RAPID IMAGE CLASSIFIER CREATION

Abstract

Image fingerprints (embeddings) are generated by an image fingerprinting model and indexed with an approximate nearest neighbors (ANN) model trained to identify the most similar fingerprint based on a subject embedding. For image matching, a score is provided that indicates a similarity between the input embedding and the most similar identified embedding, which allows for matching even when an image has been distorted, rotated, cropped, or otherwise modified. For image classification, the embeddings in the index are clustered and the clusters are labeled. Users can provide just a few images to add to the index as a labeled cluster. The ANN model returns a score and label of the most similar identified embedding for labeling the subject image if the score exceeds a threshold. As improvements are made to the image fingerprinting model, a converter model is trained to convert the original embeddings to be compatible with the new embeddings.

Inventors:	**Liao; Yihua (Palo Alto, CA), Koduri; Niranjan (Pleasanton, CA), Daryoush; Emanoel (San Jose, CA), Bryslawskyj; Jason B. (San Diego, CA), Zhang; Yi (Santa Clara, CA), Azarafrooz; Ari (Rancho Santa Margarita, CA), Xin; Wayne (Santa Clara, CA)**
Applicant:	**Netskope, Inc.** (Santa Clara, CA)
Family ID:	**1000008571960**
Appl. No.:	**19/195325**
Filed:	**April 30, 2025**

Related U.S. Application Data

parent US continuation 18451010 20230816 parent-grant-document US 12315231 child US 19195325

Publication Classification

Background/Summary

CROSS REFERENCE TO RELATED APPLICATIONS [0001] This application is a continuation of and claims priority to and the benefit of U.S. patent application Ser. No. 18/451,010, titled "IMAGE CLASSIFICATION AND ACCELERATED CLASSIFICATION TRAINING USING DEEP LEARNING IMAGE FINGERPRINTING MODELS AND INDEXED EMBEDDINGS," filed Aug. 16, 2023, the contents of which is incorporated herein by reference in its entirety for all purposes. [0002] This application is related to U.S. patent application Ser. No. 18/450,999, titled "IMAGE MATCHING USING DEEP LEARNING IMAGE FINGERPRINTING MODELS AND EMBEDDINGS," filed Aug. 16, 2023, issued May 14, 2024, as U.S. Pat. No. 11,983,955, the contents of which is incorporated herein by reference in its entirety for all purposes. [0003] This application is related to U.S. patent application Ser. No. 18/451,014, titled "IMAGE FINGERPRINTING CONVERSION BETWEEN DIFFERENT IMAGE FINGERPRINTING MODELS," filed Aug. 16, 2023, issued Mar. 4, 2025, as U.S. Pat. No. 12,243,294, the contents of which is incorporated herein by reference in its entirety for all purposes.

BACKGROUND
[0004] Images may contain private information or intellectual property. For this reason, when protecting data, it may be important to compare images being moved, stored, or shared to known images a customer wishes to protect. Historically, image matching provides very specific matching using file hashing. While it is fast and reliable for exact images, if a file has been modified at all, the hash will change and the images will not match. Some hashing methods attempt to address this issue, such as perceptual hash (pHash), but those methods cannot match images that have been rotated or cropped well. Only extremely small rotations or crops can be identified correctly with those systems. Accordingly, improved image matching is needed.
[0005] Further, when classifying images, new image classifiers require huge training data sets and large amounts of time and processing. If a user would like a classifier, the number of images needed is difficult to obtain, and often users cannot train their own classifiers. Accordingly, improved classifier training is needed.
[0006] Additionally, over the course of using models for image embedding and the like, improvements may be needed. Retraining or otherwise changing the deep learning of models make the output embeddings incompatible. The images may need to be provided to the new model for creation of embeddings. This is time consuming and costly, and sometimes the original images are not available for analysis. Accordingly, improved methods for improving models is needed.
SUMMARY
[0007] Methods, systems, and computer-readable memory devices storing instructions are described that perform enhanced image fingerprinting techniques for image matching and image classifying.

[0008] A deep learning image fingerprinting model is used to generate a semantically rich, multi-dimensional feature vector (embedding) that represents visual aspects of the input image. A set of images may be fingerprinted and used to train an approximate nearest neighbors model. The embeddings (i.e., fingerprints) from the set of images are indexed and the approximate nearest neighbors model generated to use the index to identify a most similar image embedding from the index along with a score indicating the similarity between an input image embedding and the most similar image embedding from the index. To use the image matcher, the subject image is fingerprinted and input into the approximate nearest neighbors model. If the score returned with the most similar image embedding exceeds a threshold, the images are a match. This type of matching allows for identification of a match even if an image is distorted or changed. For example, an image that has been transformed with, for example, color distortion, blurring, cropping, rotation, or the like are all identified as matches. Generally, this matching works much like a human in that if a human seeing the image would identify the image as a match, the system will identify a match.

[0009] The embeddings from the deep learning fingerprinting model may also be used for an image classifier. The embeddings are similarly indexed and an approximate nearest neighbors model generated to use the index to identify a most similar image embedding from the index. The image embeddings in the index are clustered such that similar images that should all be classified with the same label are in a cluster. Labels are applied to the embedding clusters. To use the image classifier, a subject image is fingerprinted and input into the approximate nearest neighbors model. A label of the most similar embedding and a score are returned. If the score exceeds a threshold, the image is labeled with the label. Users can easily train their own classifiers with only a few images. A user may provide as few as twenty (20) images and a class label. The images can be fingerprinted by the deep learning fingerprinting model and added to the index. An approximate nearest neighbors model is generated to use the new index, and the embeddings in the index are clustered such that the embeddings associated with the images uploaded by the user are all in a cluster and the user-defined label is applied to the cluster. When a new image is classified that belongs in the class, it will be fingerprinted, and the approximate nearest neighbors model will identify an image embedding in the user-created cluster so that the new image is classified with the user-defined label. The ability to train a classifier with such a limited number of images is a vast improvement as classifiers typically require thousands of images for training.

[0010] Over time, the deep learning image fingerprinting model may be improved, updated, or retrained. When that occurs, the image embeddings generated by the original image fingerprinting model will be different from the image embeddings generated by the new image fingerprinting model. For that reason, they cannot both be used in the same index. It may be desirous to use image embeddings from the images associated with the original image embeddings, but the images may not be available, or it may be time consuming and processing intensive to regenerate all the image embeddings with the new image fingerprinting model. An embedding converter may be used to convert image embeddings generated by the original image fingerprinting model to image embeddings compatible with image embeddings generated by the new image fingerprinting model. The embedding converter may be trained by using a new set of training images.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
[0011] In the drawings, like reference characters generally refer to like parts throughout the different views. Also, the drawings are not necessarily to scale, with an emphasis instead generally being placed upon illustrating the principles of the technology disclosed. In the following description, various implementations of the technology disclosed are described with reference to the following drawings.

[0012] FIG. **1** illustrates a network security system that includes image fingerprinting components, according to some embodiments.

[0013] FIG. **2** illustrates a data flow of data for generating an image matching system, according to some embodiments.

[0014] FIG. **3** illustrates a data flow for performing image matching, according to some embodiments.

[0015] FIG. **4** illustrates a method of image matching in a network security system, according to some embodiments.

[0016] FIG. **5** illustrates a series of images that would be matched with the image matching, according to some embodiments.

[0017] FIG. **6** illustrates a data flow for generating a user trained image classifier, according to some embodiments.

[0018] FIG. **7** illustrates a data flow for performing image classification, according to some embodiments.

[0019] FIG. **8** illustrates a data flow for performing image classification with multiple user-trained classifiers, according to some embodiments.

[0020] FIG. **9** illustrates a method of creating a user-trained image classifier and image classification in a network security system, according to some embodiments.

[0021] FIG. **10** illustrates a data flow for training an image embedding converter model, according to some embodiments.

[0022] FIG. **11** illustrates a data flow for converting image embeddings with a converter model, according to some embodiments.

[0023] FIG. **12** illustrates a method of training an image embedding converter model, according to some embodiments.

[0024] FIG. **13** illustrates an example graphical user interface for creating a user-trained classifier, according to some embodiments.

[0025] FIG. **14** illustrates an exemplary computing system, according to some embodiments.

DETAILED DESCRIPTION

[0026] Images may be important in network security systems. Images may contain private information or intellectual property. For that reason, when protecting customer data from data breaches, it may be helpful to compare image data being moved or stored to known images that customer wishes to protect. However, existing image matching systems are lacking.

[0027] In prior systems, image fingerprinting may be performed with file hashing such as md5 or sha256. File hashing is very fast and reliable as long as the file has not been modified in any way. Any change to the image or file, even flipping one bit, will completely change the hash so that it can no longer be matched to the original image file. Perceptual hash (pHash) is another image fingerprinting solution which uses image processing methods to generate a fingerprint. pHash can handle some file changes and some image distortions such as blurring and color distortion. However, since it does not use deep learning, it does not perform as well on geometric distortions such as rotation and cropping.

[0028] The present invention provides an image fingerprinting and matching system and methods that are able to identify matches despite all types of image changes including image distortions, blurring, color distortion, rotation, cropping, and the like. The deep learning used to generate the image fingerprints (embeddings) ensures the system is both robust against transformations or distortions of the image and in a manner where similar images can be queried for quickly.

[0029] This image matching system can be used in a network security system to ensure Data Loss Prevention (DLP). Images that a customer may wish to protect from possible data breaches are fingerprinted using the deep learning image fingerprinting model. The resulting fingerprints are semantically rich, multi-dimensional feature vectors called embeddings (embeddings and fingerprints are used interchangeably throughout) which are semi-unique to the fingerprinted

image. Visually similar images will produce embeddings that are closer together in the embedding space (e.g., smaller angular distance) than visually dissimilar images. This allows images which have been distorted or transformed (for example by rotation, cropping, or color distortion) to match back to their fingerprinted original. The fingerprints of protected images are added to the index of an approximate nearest neighbors model, which given a subject fingerprint from a subject image, will retrieve the most similar embedding from the index. As long as the differences between the subject fingerprint and the identified protected image fingerprint in the index are sufficiently small as represented by the score, it is considered a match. As part of DLP, customer cloud data may be scanned for images. Image fingerprint embeddings can be computed for each image with the image fingerprinting model. If the fingerprint of a scanned image matches the fingerprint of an image that the customer wishes to protect, a security policy can be enforced. For example, an administrator may be alerted.

[0030] The image fingerprinting model may further be used to generate embeddings used to classify images in an image classifier. Similar to the image matching system described above, the embeddings are added to the index of an approximate nearest neighbors model. The embeddings in the index are clustered into embedding clusters of similar images that are of the same class. Labels are applied to the embedding clusters so that all the embeddings in a given cluster have the label. When a subject image fingerprint (generated from the image fingerprinting model) is input to the approximate nearest neighbors model, it will retrieve the label of the most similar embedding from the index. As long as the differences between the subject fingerprint and the identified image fingerprint in the index are sufficiently small as represented by the score, the subject image will be classified with the label.

[0031] Image classifying may be useful in network security systems. For example, images classified with specific labels may be subject to certain security policies. A user may wish to quickly generate a new class for specific types of images, particularly images of products they may wish to protect. For example, a user may wish to protect a new design of a product and limit the ability of other users within the company to share or otherwise distribute images of the new design. However, training image classifiers has been time and resource intensive. In previous systems, the fastest way to train a new image classifier is using transfer learning.

[0032] In transfer learning, a pretrained model has been trained on a larger database of unrelated data and is either retrained or fine-tuned on the smaller dataset. Finetuning involves updating the deep learning network through backpropagation and therefore takes a considerable amount of time and is very resource intensive.

[0033] The present invention can be used to quickly train a new image classifier with a limited set of data. As few as twenty (**20**) images of a new class that one wishes to classify are fingerprinted and added to the nearest neighbors index. While as few as twenty images can be used to get acceptable results in downstream uses like image matching and image classifying, more images will improve accuracy. Instead of labeling each index entry with a unique label, all reference indices are labeled with the user-defined class name. In this way, they form a cluster in embedding space. When an image is input for classification, if it matches the new class, it is labeled with the user-defined class name (label).

[0034] The present invention is orders of magnitude faster than prior systems because it only needs to pass the new training images forward through the fingerprinting model and add the fingerprints to the index in the nearest neighbors model, avoiding backpropagation. Additionally, fine tuning image models usually requires hundreds if not thousands of images for each new class being trained. The present solution is able to be retrained with as few as twenty samples.

[0035] The deep learning image fingerprinting model used for both the image matching and image classifier may need to be retrained, improved, or otherwise changed from the original model (e.g., V1) to a new model (e.g., V2). However, if the fingerprinting model is changed in any way, updated, or retrained, to become a new model V2, the stored customer fingerprints from model V1

will no longer be compatible. The output feature vectors or embeddings of different deep learning models are usually incompatible with one another.

[0036] Usually if a model is updated or retrained, embeddings need to be recomputed using the new model. In many applications, including fingerprinting, it is impossible or undesirable for privacy reasons to access the original model inputs needed to perform this recomputation. This invention solves this problem by developing a deep learning model to convert directly the output embeddings from the old model V1 to embeddings compatible with the new model V2. The converter model is an autoencoder which has been used for input image denoising and latent variable generation but is not used as an embedding converter.

[0037] The present invention provides numerous technical advantages. For image fingerprint matching, the deep learning fingerprinting model trained using contrastive learning is very robust against transformations or distortions to the images being matched. Prior systems could not provide this level human-type matching. Using the provided system, typically if a human would identify the image as a match, the system will identify a match as well, and if a human would identify no match, the system will not match either. For training an image classifier, the resources needed to train the present classifier with a new class from a user are substantially reduced. Fewer processing cycles are needed since no loss is calculated and backpropagated. Further, memory resources are conserved because the training dataset need only be twenty images rather than the hundreds or thousands previously needed. Further, the ability to convert embeddings generated by a first model to those compatible with embeddings generated by a second model saves memory resources as original images need not be stored. In this way, data is not lost when fingerprinting models need to be updated or retrained.

[0038] Turning now to FIG. **1**, the components of system **100** that include a network security system **125** with the features of the image fingerprinting, matching, and classifying described above is depicted. System **100** includes endpoints **105**, public networks **115**, hosted services **120**, and network security system **125**.

[0039] Endpoints **105** comprise user devices including desktops, laptops, mobile devices, and the like. The mobile devices include smartphones, smart watches, and the like. Endpoints **105** may also include internet of things (IoT) devices. Endpoints **105** may include any number of components including those described with respect to computing device **1400** of FIG. **14** including processors, output devices, communication interfaces, input devices, memory, and the like, all not depicted here for clarity. Endpoints **105** may include any number of endpoints, which may be used to access content (e.g., documents, images, and the like) stored in the hosted services **120** and otherwise interact with hosted services **120**. Endpoints **105** include endpoint routing client **110**. In some embodiments, endpoint routing client **110** may be a client installed on the endpoint **105**. In other embodiments, endpoint routing client **110** may be implemented using a gateway that traffic from each endpoint **105** passes through for transmission out of a private or sub-network.

[0040] Endpoint routing client **110** routes network traffic transmitted from its respective endpoint **105** to the network security system **125**. Depending on the type of device for which endpoint routing client **110** is routing traffic, endpoint routing client **110** may use or be a virtual private network (VPN) such as VPN on demand or per-app-VPN that use certificate-based authentication. For example, for some devices having a first operating system, endpoint routing client **110** me be a per-app-VPN may be used or a set of domain-based VPN profiles may be used. For other devices having a second operating system, endpoint routing client **110** me be a cloud director mobile app. Endpoint routing client **110** can also be an agent that is downloaded using e-mail or silently installed using mass deployment tools.

[0041] Public network **115** may be any public network including, for example, the Internet. Public network **115** couples endpoints **105**, network security system **125**, hosted services **120**, such that any may communicate with any other via public network **115**. The actual communication path can be point-to-point over public network **115** and may include communication over private networks

(not shown). In some embodiments, endpoint routing client **110**, might be delivered indirectly, for example, via an application store (not shown). Communications can occur using a variety of network technologies, for example, private networks, Virtual Private Network (VPN), multiprotocol label switching (MPLS), local area network (LAN), wide area network (WAN), Public Switched Telephone Network (PSTN), Session Initiation Protocol (SIP), wireless networks, point-to-point networks, star network, token ring network, hub network, Internet, or the like. Communications may use a variety of protocols. Communications can use appropriate application programming interfaces (APIs) and data interchange formats, for example, Representational State Transfer (REST), JavaScript Object Notation (JSON), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Java Message Service (JMS), Java Platform Module System, and the like. Additionally, a variety of authorization and authentication techniques, such as username/password, Open Authorization (OAuth), Kerberos, SecureID, digital certificates and more, can be used to secure communications.

[0042] Hosted services **120** can be cloud computing and storage services, financial services, e-commerce services, or any type of applications, websites, or platforms that provide cloud-based storage or web services. Hosted services **120** can be referred to as cloud services, cloud applications, cloud storage applications, cloud computing applications, or the like. Hosted services **120** provide functionality to users that can be implemented in the cloud and that can be the target of data loss prevention (DLP) policies, for example, logging in, editing documents, downloading data, reading customer contact information, entering payables, deleting documents, and the like. Hosted services **120** can be a network service or application, or can be web-based (e.g., accessed via a URL) or native, such as sync clients. Examples include software-as-a-service (SaaS) offerings, platform-as-a-service (PaaS) offerings, and infrastructure-as-a-service (IaaS) offerings, as well as internal enterprise applications that are exposed via URLs. While only one hosted services **120** is depicted in FIG. **1**, any number of hosted services may be available and included in system **100**. Hosted services may be sanctioned (e.g., those that a company provides for employee use and of which the company's information technology (IT) department is aware) or unsanctioned (e.g., those a company is not aware of or otherwise are not authorized for use by the company).

[0043] Network security system **125** may provide network security services to endpoints **105**. Endpoint routing client **110** may route traffic from the endpoints **105** to network security system **125** to enforce security policies including DLP policies. Network security system **125** may be one or more computing systems such as computing device **1400** as described with respect to FIG. **14**. Network security system includes image fingerprinter **130**, index and approximate nearest neighbors (ANN) model generator **132**, image matcher **134**, image classifier **136**, classifier trainer **138**, security policy enforcer **140**, embedding converter **142**, converter trainer **144**, user interface **146**, model data store **148**, embedding data store **150**, security policy data store **152**, training data store **154**, and index data store **156**. The modules of network security system **125** may be implemented in hardware or software and need not be divided up in precisely the same blocks as shown in FIG. **1**. Some of the modules can also be implemented on different processors or computers or spread among any number of different processors or computers. In addition, in some embodiments, modules may be combined, operated in parallel, or in a different sequence than that shown without affecting the functions achieved and without departing from the spirit of this disclosure. Also, as used herein, the term "module" can include "sub-modules," which themselves can be considered to constitute modules. The term module may be interchanged with component and neither term requires a specific hardware element but rather indicates a device or software that is used to provide the described functionality. The modules (i.e., shown as blocks and data stores) in network security system **125** may, in some embodiments, also be thought of as flowchart steps in a method. In some embodiments, a software module need not have all its code disposed contiguously in memory (e.g., some parts of the code can be separated from other parts of the code with code from other modules or other functions disposed in between).

[0044] Image fingerprinter **130** uses a deep learning image fingerprinting model, stored in model data store **148**, to generate image fingerprints, also called image embeddings throughout. A single image fingerprinting model is used for each of the image fingerprints so they are compatible throughout the system with each of the described components. Image fingerprinter **130** receives an image for which an image fingerprint is desired. Image fingerprinter **130** may perform functions on the image data to provide the image data as input to the image fingerprinting model. Image fingerprinter **130** provides the image data to the image fingerprinting model and receives, as output, an image fingerprint (i.e., an image embedding). Image embeddings are semantically rich multi-dimensional feature vectors. For example, the feature vectors may have five hundred twelve (**512**) dimensions. Image fingerprinter **130** provides the image embeddings once received from the deep learning image fingerprinting model.

[0045] The deep learning image fingerprinting model may be a neural network (NN)-based image encoder model (e.g., a convolutional neural network (CNN)). The image fingerprinting model may be trained using contrastive training framework such as, for example, simCLR. Image models may be trained in an unsupervised fashion by taking each sample image and transforming (augmenting) it in a way that does not destroy the semantic information in the image. For example, random rotations, random color distortions, and random crops (e.g., with the crop size restricted to a large area of the original image) may be used to augment the sample image. These augmented sample images are used to train the image fingerprinting model. The image fingerprinting model takes image data as inputs and, in return, outputs an image embedding (e.g., a multi-dimensional feature vector). During training, the image fingerprinting model learns to maximize the similarity of the image embedding from the original image to that of the augmented image while simultaneously minimizing the similarity to all other images in the batch of training images. A loss function is used for training, which may be, for example, essentially the softmax of the cosine similarity of all pairs of image representations including the original and augmented original image. After training, the image fingerprinting model can transform images into a generic and semantically rich vector representation (i.e., the image embeddings) without the need of supervised training. The image embeddings can then be used in a downstream model such as the described image matching or image classifying.

[0046] Index and ANN model generator **132** generates an index by indexing image embeddings and generates an ANN model to use with the index. Index and ANN model generator **132** may store the indexes in index data store **156** and the ANN models in model data store **148**. In some embodiments, index data store **156** is not used because indexes are not stored. In such embodiments, an index is built dynamically in memory on demand when an index is needed. For example, when an index is needed for image matching or image classification, the relevant image embeddings are accessed from embedding data store **150** and the index is generated and used by the ANN model. Image embeddings may be received from image fingerprinter **130** and used to generate a new index or may be added to an existing index. When adding to an existing index, the previous index may be stored and a new index generated including both the old image embeddings and the newly added image embeddings. When the index is dynamically generated, the new image embeddings are stored with the old image embeddings to generate the new index. Any time an index is newly generated or modified, or the stored embeddings used to dynamically generate the index are changed, a new ANN model is generated to use with the index because the ANN model and associated index are linked to be used together. Each index creates an embedding space, and the ANN model is trained to identify, given a subject embedding, the closest reference image embedding in the embedding space. Embedding similarity is measured by angular distance. An example equation is $D=\sqrt{2-2*\cos(x\_r, x\_s)}$ where x_r is the reference embedding, and x_s is the subject embedding. In some embodiments, the Annoy framework is used which leverages the approximate nearest neighbors algorithm to retrieve similar vectors from the index.

[0047] For image matching, index and ANN model generator **132** generates the ANN model such

that it is trained to return the image embedding of the most similar image and a score indicating the similarity. For example, the score may be a distance calculation normalized to a value between zero and one hundred (0-100), which may be thought of as a percent of similarity. For example, the score may be an angular distance representing the differences between the two embeddings. In such an example, a maximum distance is an angle of Pi divided by two (Pi/2) and a minimum distance is an angle of zero (0). A zero angle results from one hundred percent (100%) similarity, and a Pi divided by two (Pi/2) angle results from zero percent (0%) similarity. An example equation is (Pi/2−acos(1−AngularDistance*AngularDistance/2))/(Pi/2). In other words, a score of seventy (70) may indicate the images of the corresponding image embeddings may be thought of as approximately seventy (70) percent similar. While the score does not exactly correspond to a percentage of similarity, it provides a meaningful value indicating a likelihood that the images match. Accordingly, the smaller the value, the less likely the images match.

[0048] For image classification, index and ANN model generator **132** clusters the image embeddings in the index into embedding clusters. Index and ANN model generator **132** labels the embedding clusters with a corresponding label such that each image embedding in the embedding cluster is labeled with the label for the cluster. Index and ANN model generator **132** generates the ANN model to return the label of the image embedding of the most similar image and a score indicating the similarity. For example, the score may be an angular distance normalized to a value between zero and one hundred (0-100), which may be thought of as a percent of similarity. In other words, a score of eighty-five (85) may be thought of as representing that the images of the corresponding image embeddings are approximately eighty-five (85) percent similar.

[0049] Image matcher **134** may be used to identify matching images. For example, a customer may wish to protect certain images or designs. Image matcher **134** may be used to identify images that match the protected images by scanning documents stored in hosted services **120** or may identify images when users engage with the image or document containing the image by uploading, downloading, sharing, moving, or otherwise interacting with the image on endpoints **105**. Image matcher **134** may receive the image and transmit the image to image fingerprinter **130** to obtain the image embedding. Image matcher **134** may then access the ANN model stored in model data store **148** used for image matching and provide the image embedding as an input to the ANN model. In some embodiments, multiple ANN models may be stored in model data store **148**. Image matcher **134** may choose the most recent image matching ANN model, in some embodiments. In some embodiments, image matcher **134** may choose the ANN model from model data store **148** based on the set of images that the subject image is to be matched against. For example, image data sets may be named when uploaded for matching, and the user may select which image data set should be matched against when providing an image. In some embodiments, image matcher **134** may use multiple ANN models to check multiple image data sets for a match. Image matcher **134** may receive the most similar image embedding and the score in return. Image matcher **134** may include a threshold value to compare the score with. The threshold value may be, for example eighty-five (85), indicating generally that the image embedding and the most similar image embedding are approximately 85% similar. A threshold value of eighty-five (85) proved robust enough to identify most matches while limiting false positives in the tested system, however any appropriate threshold value may be used. When the score exceeds, or is equal to, the threshold value, the image may be identified as a match. When the score is below the threshold value, the image may be identified as not a match. Image matcher **134** may apply metadata to the image to indicate whether it is a match or not in some embodiments. Image matcher **134** may transmit the image and an indication of whether or not the image matches a protected image to security policy enforcer **140** for enforcing a security policy based on whether or not the image is a match (i.e., based on the score).

[0050] Image classifier **136** may be used to classify images and may be used to apply one or more labels to images based on the classifications. For example, certain types of images may be protected, or a client may otherwise want to label the images. Image classifier **136** may be used to

identify classes an image belongs to. Image classifier **136** may identify images for classifying by scanning documents stored in hosted services **120** or may identify images when users engage with the image or document containing the image by uploading, downloading, sharing, moving, or otherwise interacting with the image on endpoints **105**. Image classifier **136** may receive the image and transmit the image to image fingerprinter **130** to obtain the image embedding. Image classifier **136** may then access one or more ANN models stored in model data store **148** used for image classifying and provide the image embedding as an input to the ANN models. Image classifier **136** may select the most recent image classifying ANN model. In some embodiments, image classifier **136** may select all ANN models having a classifier that may be relevant for the given image to be classified against. For example, a user may train multiple classifiers using classifier trainer **138**, and each of the relevant generated ANN models may be used by image classifier **136**. Image classifier **136** may receive the label of the most similar image embedding and the score in return. Image classifier **136** may include a threshold value to compare the score with. The threshold value may be, for example eighty-five (85), indicating generally that the image embedding and the most similar image embedding are approximately 85% similar. A threshold value of eighty-five (85) proved robust enough to identify most matches while limiting false positives in the tested system, however any appropriate threshold value may be used. When the score exceeds, or is equal to, the threshold value, the returned label may be applied to the subject image. When the score is below the threshold value, the label is not applied to the image. Image classifier **136** may apply metadata to the image to indicate the applied label. In some embodiments, if multiple ANN models are used, any label returned with a score equal to or exceeding the threshold value may be applied to the subject image, resulting in an image that has multiple labels. For example, if a first label is returned from a first ANN model with a score of eighty-seven (87) and a second label is returned from a second ANN model with a score of ninety-two (92), both the first label and the second label may be applied to the subject image. In some embodiments, only the label having the highest score that is equal to or exceeds the threshold value is applied. For example, if a first label is returned from a first ANN model with a score of eighty-seven (87) and a second label is returned from a second ANN model with a score of ninety-two (92), the second label may be applied to the subject image. If no labels are returned with a score exceeding (or equal to) the threshold value, a default label (e.g., "other") may be applied to the image. Image classifier **136** may transmit the image and an indication of applied labels to security policy enforcer **140** for enforcing a security policy based on the image classification (i.e., based on the score).

[0051] Classifier trainer **138** is used to quickly train a new classifier, which can allow a user to specify and create classifiers based on their own information quickly and easily. Classifier trainer **138** does not use the expensive retraining of the deep learning image fingerprinting model and does not need large amounts of data to fine tune an end-to-end CNN model. Classifier trainer **138** uses user interface **146** to obtain images and a class label from the user. The user can provide a set of as few as twenty (20) images and a class label. While as few as twenty images can be used and results in acceptable results from image classifier **136**, more user images will provide better accuracy. For example, forty (40) images provide better accuracy, and eighty (80) images provide even better accuracy. Less than one hundred (100) images are needed to provide solid accuracy. Classifier trainer **138** inputs the images to image fingerprinter **130** to get image embeddings for the training images. Classifier trainer **138** uses index and ANN model generator **132** to add the new training embeddings to a base index. When generating this user-defined classifier (also called Train Your Own Classifier (TYOC)), a base index that may include negative image embeddings classified with a default label (e.g., "other") may be used. In some embodiments the base index may also include other general classes. In some embodiments, the user may be able to add multiple classes by adding to the prior user-defined classifier. In some embodiments, the user may also be able to add additional negative image embeddings. In some embodiments, if a user uploads an image with an image embedding already in the index, the image embedding in the index may be reclassified into

the embedding cluster with the new user-defined label. In other embodiments, the additional label may be added, and the image embedding may be in more than one embedding cluster. In some embodiments, a user-defined negative image may overwrite other labels applied to the image previously.

[0052] Classifier trainer **138** is fast and efficient because it does not need retraining of the deep learning fingerprinting model. Instead, the training images (e.g., approximately twenty to one hundred training images) are added to the index. Instead of labeling each index entry (i.e., image embedding) with a unique label, all added image embeddings are labeled with the user-defined label (i.e., class name). In this way, the user-provided training images form a cluster in embedding space and are easily used to classify new image embeddings using image classifier **136** without extensive retraining of deep learning models or large training data sets.

[0053] Security policy enforcer **140** may perform security policy enforcement on images based on results from image classifier **136**, image matcher **134**, or both. Further, security policy enforcer **140** may perform security policy enforcement on all other messages transmitted to network security system **125**. The security policies may include blocking the activity (e.g., moving, sharing, deleting, adding, or the like the information or documents) included in the messages. In other examples, the security policies may include allowing the activity, notifying an administrator of the activity, or providing a notice to the user at the associated endpoint **105**.

[0054] Regarding security policies enforced by security policy enforcer **140** for image matcher **134**, a client may upload specific images for matching for use in DLP. Image matcher **134** may perform image matching for images that are identified based on user activity as messages are passed through network security system **125**. For images matched using image matcher **134** based on user activity, security policy enforcer **140** may block the activity, for example. In some embodiments, security policy enforcer **140** may notify an administrator of the activity in addition to or instead of blocking the activity. In some embodiments, client data stored in hosted services **120** may be scanned to identify matching images. When an image is identified in hosted services **120**, security policy enforcer **140** may notify an administrator, for example. In some embodiments, security policy enforcer **140** may delete the image from hosted services **120**. Security policies may be set to perform any security action based on the configuration of the security policy.

[0055] For security policies enforced by security policy enforcer **140** from image classifier **136**, a client may train a classifier for classifying images for use in DLP. Security policy enforcer **140** may use the image labels to determine a security policy to enforce. In some embodiments, when the user trains a classifier, the data stored in hosted services **120** can be scanned to identify images to classify. Depending on whether labels are applied to the images, security policy enforcer **140** may enforce a security policy to, for example, notify an administrator of the image, delete the image, or both. Further, images may be classified by image classifier **136** based on user activity intercepted by network security system **125**. The activity may be blocked, an administrator notified, or both based on the image labels (i.e., classifications).

[0056] Embedding converter **142** may be trained by converter trainer **144** to convert image embeddings from a first deep learning image fingerprinting model to a second deep learning image fingerprinting model (e.g., a first version of the image fingerprinting model to a second version of the image fingerprinting model). For image fingerprinting used for matching and classifying images, it may be undesirable to request reference images from customers more than once. However, it may be advantageous to retrain the underlying deep learning fingerprinting model with real production data in order to improve efficacy, for example. Once the underlying deep learning fingerprinting model is retrained, even when starting with the weights and biases of the old model, an entirely new model is created (e.g., version 2 or V2). Images that are fingerprinted with the first model result in image embeddings that are different and incompatible with the image embeddings generated by the second model. Accordingly, the image embeddings generated by the first model cannot be indexed with the image embeddings generated by the second model because the results

will not be accurate. However, the image embeddings generated by the first model may still be needed. For example, they may be image embeddings of the customer's protected information used in an index for image matcher **134**. Since the original images may not be available, converting the image embeddings from the first model to be compatible with image embeddings generated by the second model is helpful. Accordingly, converter trainer **144** is used to train a deep learning converter model to convert the image embeddings.

[0057] The deep learning converter model uses a deep learning architecture similar to an autoencoder. The input of the model is the image embeddings generated by the first deep learning image fingerprinting model. The output is a converted image embedding compatible with image embeddings generated by the second deep learning image fingerprinting model. Each block of the deep learning converter model consists of a fully connected layer with dropout and Batch Normalization. As in a typical autoencoder, the layers become progressively smaller and then progressively larger. This creates an information bottleneck which forces the converter model to maximize the information content of the image embeddings.

[0058] Converter trainer **144** uses a set of training images to train the converter model. The training images may be a different set of training images not used to train either the first image fingerprinting model or the second image fingerprinting model. Converter trainer **144** sends the training images to image fingerprinter **130** to generate a first set of image embeddings using the first image fingerprinting model and a second set of image embeddings using the second image fingerprinting model. Converter trainer **144** passes the first set of image embeddings through the converter model and obtains converted image embeddings. Converter trainer **144** compares the converted image embeddings to the associated image embedding from the second set of image embeddings generated by the second image fingerprinting model. Converter trainer **144** may use Mean Square Error to compare the image embeddings. Converter trainer **144** calculates the loss (e.g., the Mean Square Error) and backpropagates the loss into the converter model to train the converter model. When done with the training, the converter model is trained to produce image embeddings compatible with image embeddings generated by the second image fingerprinting model from image embeddings generated by the first image fingerprinting model. The converter model may be stored in model data store **148**.

[0059] Once trained, embedding converter **142** may be used to convert all needed image embeddings generated by the first image fingerprinting model to those that are compatible with image embeddings generated by the second image fingerprinting model. Embedding converter may access the image embeddings generated by the first image fingerprinting model and pass them through the converter model to generate converted image embeddings. Embedding converter **142** may also take the sets of converted image embeddings that were used in a given index and provide them to index and ANN model generator **132** to generate new indexes and ANN models. The new indexes and ANN models can be used for new image embeddings generated by the second (i.e., updated) index fingerprinting model to identify matches and to classify images using image matcher **134** and image classifier **136**, respectively.

[0060] User interface **146** may be used to allow a user to interface with network security system **125** to train a classifier, upload images for matching, set security policies, or the like. A user on an endpoint **105** may access network security system **125** via the user interface **146**, which may provide graphical user interfaces for display on the endpoint **105**. An example graphical user interface **1300** is depicted in FIG. **13**. Endpoint **105** may access user interface **146** via a web browser, an installed app, or the like.

[0061] Model data store **148** may store models including deep learning image fingerprinting models, converter models, and the like. Model data store **148** may also store ANN models generated by index and ANN model generator **132**. Model data store **148** may be a portion of a larger data store that stores other items or only used for storing models.

[0062] Embedding data store **150** may store image embeddings generated by image fingerprinting

model. Some image embeddings may be stored for later use including the image embeddings used to generate any of the described indexes. The image embeddings may be used later to generate new indexes of different groupings of the image embeddings, for example. While some image embeddings may be stored, not every image embedding may be stored in some embodiments.

[0063] Security policy data store **152** may store security policies used by security policy enforcer **140**. The security policies may include those used for image matching and image classification as well as any other security policies used for enforcing security policies by network security system. While image security is discussed throughout, other security policies including other DLP security policies may be enforced with network security system **125**, and image security may only be a portion of the security provided by network security system **125**.

[0064] Training data store **154** may store image data sets that are used for training deep learning models including image fingerprinting models and converter models. Training data store **154** may store historical information about training data sets without maintaining the entire image data set over time. In some embodiments, training data store **154** may be used for staging image data sets for training, and the images may be deleted after image fingerprint embeddings are generated or training is complete. In some embodiments, image data sets that are used by classifier trainer **138** to train classifiers are stored in training data store **154**. In some embodiments, image data sets used by classifier trainer **138** are stored only for staging and deleted after image fingerprint embeddings are generated or training is complete. In some embodiments, training data store **154** is not used and other temporary storage is used for staging all image data sets.

[0065] Index data store **156** may store indexes generated by index and ANN model generator **132**. When replacement indexes are generated, such as, for example, when a new classifier is trained by a user, the old index may be retained in index data store **156** in some embodiments. In other embodiments, the old index may be deleted.

[0066] Model data store **148**, embedding data store **150**, security policy data store **152**, training data store **154**, and index data store **156** are described as individual data stores, but may be combined in any combination.

[0067] The various functionalities of the modules of network security system **125** will be described in use in the descriptions of the following figures.

[0068] FIG. **2** illustrates a data flow **200** for generating an index and training an ANN model for use by image matcher **134**. Training images **205** may be obtained from training data store **154**. Training images **205** may be input into image fingerprinter **130**. Image fingerprinter **130** inputs training images **205** into deep learning image fingerprinting model **210** to generate image fingerprint embeddings **215** from the images. Image fingerprint embeddings **215** may be stored in embeddings data store **150**. Image fingerprint embeddings **215** may also be input to index and ANN model generator **132**. Index and ANN model generator **132** may index the image fingerprint embeddings **215** into ANN index **225** and train approximate nearest neighbors (ANN) model **220** to use ANN index **225** to identify the most similar image embedding from image fingerprint embeddings **215** to an image embedding of a subject image embedding input into ANN model **220**. Once generated and trained, index and ANN model generator **132** stores ANN model **220** into model data store **148** and ANN index **225** in index data store **156**. They are now ready to be used by image matcher **134**.

[0069] FIG. **3** illustrates a data flow **300** for using the ANN model **220** and ANN index **225** by image matcher **134**. Image matcher **134** receives subject image **305** for matching against image fingerprint embeddings **215**. Image matcher **134** transmits subject image **305** to image fingerprinter **130**. Image fingerprinter **130** inputs subject image **305** into deep learning image fingerprinting model **210** to generate subject image fingerprint embedding **310**. In some embodiments, image matcher **134** transmits subject image fingerprint embedding **310** to embeddings data store **150** for storage. Image matcher **134** inputs subject image fingerprint embedding **310** into ANN model **220**, which is trained to identify the most similar image fingerprint embedding in ANN index **225** to

subject image fingerprint embedding **310** and generate a score representing the similarity. The score may represent a proportional mapping of an angular distance between the most similar image fingerprint embedding and subject image fingerprint embedding **310**. The score, therefore, represents the differences between the two embeddings. ANN model **220** outputs the most similar image fingerprint embedding and score **315**. Image matcher **134** uses match determiner **320** to determine whether subject image **305** is a match to the image used to generate the most similar image fingerprint embedding based on the score. For example, match determiner **320** may compare the score to a threshold value. If the score meets or exceeds the threshold value, match determiner **320** may identify subject image **305** as a match. If the score is below the threshold value, match determiner **320** may determine subject image **305** is not a match. Image matcher **134** may provide the determination from match determiner **320** with the subject image **305** to security policy enforcer **140**. Security policy enforcer **140** may enforce a security policy on subject image **305** based on whether subject image **305** is a match (i.e., based on the score). For example, if subject image **305** is a match, security policy enforcer may apply a security policy relevant to images that are matches to the images for which image embeddings are in ANN index **225** as well as one or more other security policies. If subject image **305** is not a match, security policy enforcer **140** may not apply the security policy relevant to matches but instead a different one, others of the one or more other security policies that would be applied if subject image **305** were a match, or a combination.

[0070] FIG. **4** illustrates a method **400** for image matching using a network security system such as network security system **125**. Method **400** begins at **405** with generating image fingerprint embeddings using the image fingerprinting model. For example, image fingerprinter **130** may input training images into deep learning image fingerprinting model **210** to generate image fingerprint embeddings **215**.

[0071] At **410**, method **400** continues with generating an index of the image fingerprint embeddings and an approximate nearest neighbors model trained to use the index. For example, image fingerprinter **130** may send image fingerprint embeddings **215** to index and ANN model generator **132** to generate ANN index **225** and ANN model **220**.

[0072] At **415**, method **400** continues with providing a subject image as input to the image fingerprinting model. For example, image matcher **134** may receive subject image **305** and input it into image fingerprinter **130**, which uses deep learning image fingerprinting model **210** to generate subject image fingerprint embedding **310**.

[0073] At **420**, method **400** continues with receiving a subject image fingerprint embedding as output from the image fingerprinting model. For example, image matcher **134** may receive the subject image fingerprint embedding **310** from deep learning image fingerprinting model **210**.

[0074] At **425**, method **400** continues with providing the subject image fingerprint embedding as input to the approximate nearest neighbors model trained to retrieve a most similar image fingerprint embedding from the index and generate a score indicating a similarity of the subject image fingerprint embedding to the most similar fingerprint embedding. For example, image matcher **134** inputs subject image fingerprint embedding **310** to the ANN model **220**, which is trained to retrieve a most similar image fingerprint embedding from the ANN index **225** and generate a score.

[0075] At **430**, method **400** continues with analyzing the score received from the approximate nearest neighbors model. For example, match determiner **320** analyzes the score from ANN model **220** to determine whether subject image fingerprint embedding **310** is a match to the most similar fingerprint embedding. In some embodiments, the score is compared to a threshold value. If the score meets or exceeds the threshold value, subject image fingerprint embedding **310** is determined a match, otherwise subject image fingerprint embedding **310** is determined no match.

[0076] At **435**, method **400** continues with performing a security action based on the score. For example, image matcher **134** may send subject image **305** and an indication of whether it is a match

to any images used to generate ANN index **225** to security policy enforcer **140**. Security policy enforcer **140** may apply security policies to subject image **305** based on whether it is a match or not (i.e., based on the score). If subject image **305** is a match, different security policies may be applied than if it is not a match. In some embodiments, if subject image **305** is a match, a notification may be sent to an administrator, an activity (e.g., moving, sharing, uploading, downloading, or the like) being performed on subject image **305** may be blocked, or both. If subject image **305** is not a match, no message is sent, the activity is not blocked, or both.

[0077] The steps of method **400** may be performed in a different order than that described including some steps being performed in parallel, in some embodiments. Further, method **400** may include more or fewer steps than those described.

[0078] FIG. **5** illustrates example images having differing transformations that may be identified using image matcher **134**. Image **505** may be the original image. Image **510** may be a transformation of image **505** in which image **505** is flipped. Image **515** is a cropping of image **505**. Image **520** is image **505** transformed with a rotation. Each of image **510**, image **515**, and image **520** may be identified as a match to image **505** using image matcher **134**. For example, image **505** may be one of the images in training data **205**. An image embedding may be generated as one of image fingerprint embeddings **215** and used to generate ANN model **220** and ANN index **225**. When any of image **510**, image **515**, or image **520** is input into image matcher **134** as a subject image, image fingerprinter **130** uses deep learning image fingerprinting model **210** to generate subject image fingerprint embedding **310**, and ANN model **220** will identify the image embedding associated with image **505** from ANN index **225** as a match and provide a score. Match determiner **320** will identify image **505** as a match to any of image **510**, image **515**, and image **520**.

[0079] Advantageously, image matcher **134** can identify transformations including cropping, rotation, color distortions, blurring, and the like.

[0080] FIG. **6** illustrates a data flow **600** for training a classifier using a small data set quickly and efficiently without resorting to large data sets and retraining deep learning models like the deep learning image fingerprinting model. User provided training images **605** may be a small set of images including as few as twenty (20) images, though more images may be used to improve accuracy. User provided training images **605** may be obtained via a graphical user interface such as graphical user interface **1300** described with respect to FIG. **13** and may be stored in training data store **154** upon receipt. Training images **605** may be input into image fingerprinter **130**. Image fingerprinter **130** inputs training images **605** into deep learning image fingerprinting model **210** to generate image fingerprint embeddings **610** from the images. Image fingerprint embeddings **610** may be stored in embeddings data store **150**. Image fingerprint embeddings **610** may also be added to index **615**. Index **615** may be a base index including negative image embeddings labeled with a default label (e.g., "other"), for example. Classifier trainer **138** may input image embeddings from index **615** plus the image fingerprint embeddings **610** to index and ANN model generator **132**. Index and ANN model generator **132** may index the combined image fingerprint embeddings into ANN index **625** and train approximate nearest neighbors (ANN) model **620** to use ANN index **625** to identify the most similar image embedding from the combined image embeddings to an image embedding of a subject image embedding input into ANN model **620**. Once ANN index **625** is generated and ANN model **620** is generated and trained, classifier trainer **138** generates embedding clusters (**630**) in ANN index **625** and applies labels to the embedding clusters **635**. Specifically, image fingerprint embeddings **610** are clustered into a single embedding cluster and the user-defined label, provided with training images **605**, is applied to the embedding cluster to form the cluster in embedding space. Other image embeddings in ANN index **625** are similarly clustered and labeled. Classifier trainer stores ANN model **620** into model data store **148** and ANN index **625** in index data store **156**. They are now ready to be used by image classifier **136**. Notably, generating ANN index **625** and ANN model **620** to use ANN index **625** is magnitudes of order faster than training a deep learning model, and the accuracy of the resulting classifications is comparable to an

end-to-end deep learning classifier.

[0081] FIG. **7** illustrates a data flow **700** for using the ANN model **620** and ANN index **625** by image classifier **136**. Image classifier **136** receives subject image **705** for classifying based on the classifier trained with image fingerprint embeddings **610** (i.e., ANN index **625** and ANN model **620**). Image classifier **136** transmits subject image **705** to image fingerprinter **130**. Image fingerprinter **130** inputs subject image **705** into deep learning image fingerprinting model **210** to generate subject image fingerprint embedding **710**. In some embodiments, image classifier **136** transmits subject image fingerprint embedding **710** to embeddings data store **150** for storage. Image classifier **136** inputs subject image fingerprint embedding **710** into ANN model **620**, which is trained to identify the most similar image fingerprint embedding in ANN index **625** to subject image fingerprint embedding **710** and generate a score representing the similarity. The score may represent a normalized value of an angular distance between the most similar image fingerprint embedding and subject image fingerprint embedding **710**. ANN model **620** outputs the label of the most similar image fingerprint embedding and score **715**. Image classifier **136** uses labeling **720** to determine whether subject image **705** matches the image used to generate the most similar image fingerprint embedding closely enough based on the score to apply the label or classification to subject image **705**. For example, labeling **720** may compare the score to a threshold value. If the score meets or exceeds the threshold value, labeling **720** may label subject image **705** with the returned label (i.e., class name). If the score is below the threshold value, labeling **720** may determine subject image **705** does not belong in the class and not apply the label. In some embodiments, a default label (e.g., "other") may be applied to subject image **705** if the score is below the threshold value. Image classifier **136** may provide subject image **705** with its designated label, based on the score, to security policy enforcer **140**. Security policy enforcer **140** may enforce a security policy on subject image **705** based on the label, if any, applied to subject image **705** (i.e., based on the score). For example, if subject image **705** is labeled with the user-defined label, security policy enforcer may apply a security policy relevant to images that are classified in the user-defined class as well as one or more other security policies. If subject image **705** is labeled with the default label, security policy enforcer **140** may not apply the security policy relevant to the user-defined class but instead a different one, others of the one or more other security policies that would be applied if subject image **705** were in the user-defined class, or a combination.

[0082] FIG. **8** illustrates a data flow **800** for using the ANN model **620** and ANN index **625** as well as a second ANN model **820** and ANN index **825** by image classifier **136**. For example, a user may train more than one classifier as described in FIG. **6**. In some embodiments, the additional class may be added to the first index to generate a single ANN model and ANN index that may apply the first user-defined label or the second user-defined label depending on the input image. However, a user may wish to train image classifiers separately, allowing for labeling to occur differently depending on the resulting classifications.

[0083] Image classifier **136** receives subject image **805** for classifying. Image classifier **136** transmits subject image **805** to image fingerprinter **130**. Image fingerprinter **130** inputs subject image **805** into deep learning image fingerprinting model **210** to generate subject image fingerprint embedding **810**. In some embodiments, image classifier **136** transmits subject image fingerprint embedding **810** to embeddings data store **150** for storage. Image classifier **136** inputs subject image fingerprint embedding **810** into ANN model **620**, which is trained to identify the most similar image fingerprint embedding in ANN index **625** to subject image fingerprint embedding **810** and generate a score representing the similarity. Additionally, image classifier **136** inputs subject image fingerprint embedding **810** into ANN model **820**, which is trained to identify the most similar image fingerprint embedding in ANN index **825** to subject image fingerprint embedding **810** and generate a score representing the similarity. ANN model **620** and ANN index **625** may be a first user-defined classifier, and ANN model **820** and ANN index **825** may be a second user-defined classifier, for example. The scores output from each may represent a normalized value of an

angular distance between the most similar image fingerprint embedding and subject image fingerprint embedding **810**. ANN model **620** outputs the label of the most similar image fingerprint embedding and score **815**. ANN model **820** outputs the label of the most similar image fingerprint embedding and score **830**. Image classifier **136** uses labeling **835** to determine whether subject image **805** matches the image used to generate the most similar image fingerprint embedding from ANN model **620** and ANN index **625** closely enough based on the score to apply that label or classification to subject image **805**. Image classifier **136** also uses labeling **835** to determine whether subject image **805** matches the image used to generate the most similar image fingerprint embedding from ANN model **820** and ANN index **825** closely enough based on the score to apply that label or classification to subject image **805**. For example, labeling **835** may compare each score to a threshold value. In some embodiments, each ANN model may have a different threshold value for comparison, and accordingly each score is compared with its respective threshold value. If one score meets or exceeds the relevant threshold value, labeling **835** may label subject image **805** with the returned label (i.e., class name) having a score that exceeds the threshold. If each score is below the relevant threshold value, labeling **835** may determine subject image **805** does not belong in either class and not apply either label. In some embodiments, a default label (e.g., "other") may be applied to subject image **805** if each score is below the threshold value. Each score may meet or exceed the relevant threshold value in some scenarios. In that case, in some embodiments, labeling **835** may be configured to apply both labels to subject image **805**. In other embodiments, labeling **835** may be configured to apply the label having the highest score to subject image **805**. Image classifier **136** may provide subject image **805** with its designated label or labels, based on the scores, to security policy enforcer **140**. Security policy enforcer **140** may enforce a security policy on subject image **705** based on the label, if any, applied to subject image **805** (i.e., based on the score). For example, if subject image **805** is labeled with one or more user-defined labels, security policy enforcer **140** may apply security policies relevant to images that are classified in the user-defined classes as well as one or more other security policies. If subject image **805** is labeled with the default label, security policy enforcer **140** may not apply the security policies relevant to the user-defined classes but instead different ones, others of the one or more other security policies that would be applied if subject image **805** were in the user-defined class, or a combination.

[0084] FIG. **9** illustrates a method **900** for image classification using a network security system such as network security system **125**. Method **900** begins at **905** with generating first image fingerprint embeddings using the image fingerprinting model. For example, image fingerprinter **130** may input a set of base training images into deep learning image fingerprinting model **210** to generate image fingerprint embeddings.

[0085] At **910**, method **900** continues with generating a first index of the first image fingerprint embeddings including clustering subsets of the first image fingerprint embeddings into embedding clusters and generate an associated first approximate nearest neighbors model trained to use the first index. For example, image fingerprinter **130** may send the first generated image fingerprint embeddings to index and ANN model generator **132** to generate a base index (index **615**) and an associated ANN model. Classifier trainer **138** may generate embedding clusters (**630**) of index **615**.

[0086] At **915**, method **900** continues with applying a label to each of the embedding clusters. For example, classifier trainer **138** may apply labels to the embedding clusters (**635**).

[0087] At **920**, method **900** continues with receiving a set of images selected by a user and a user-defined label for the set of images. For example, a user may provide a set of training images **605** and a user-defined label using graphical user interface **1300**. Classifier trainer **138** may receive the user provided training images **605** and the user-defined label.

[0088] At **925**, method **900** continues with generating second image fingerprint embeddings using the image fingerprinting model. For example, classifier trainer **138** inputs the user provided training images **605** into image fingerprinter **130**. Image fingerprinter **130** provides the user

provided training images **605** to deep learning image fingerprinting model **210** to generate image fingerprint embeddings **610**.

[0089] At **930**, method **900** continues with adding the second image fingerprint embeddings to the first index to generate a second index and an associated second approximate nearest neighbors model. For example, classifier trainer **138** adds image fingerprint embeddings **610** to the base index, index **615**. Classifier trainer **138** provides the combined set of image fingerprint embeddings and provides them as input to index and ANN model generator **132**. Index and ANN model generator **132** generates a new index (ANN index **625**) of the combined image embeddings and an associated approximate nearest neighbors model (ANN model **620**).

[0090] At **935**, method **900** continues with clustering the second image fingerprint embeddings in the second index and apply the user-defined label. For example, classifier trainer **138** generates embedding clusters (**630**) and applies labels to the embedding clusters (**635**). For example, the user-defined label is applied to each of the image fingerprint embeddings **610** to generate the cluster in embedding space.

[0091] The steps of method **900** may be performed in a different order than that described including some steps being performed in parallel, in some embodiments. Further, method **900** may include more or fewer steps than those described.

[0092] FIG. **10** illustrates a data flow **1000** for converter trainer **144** to train a converter model **1020** to convert image embeddings generated by a first deep learning image fingerprinting model **1010** to image embeddings compatible with image embeddings generated by a second deep learning image fingerprinting model **1030**. In some cases, for example, a first deep learning image fingerprinting model **1010**, which may be a first version, may need to be retrained or otherwise updated, which results in the second deep learning image fingerprinting model **1030**. Once modified, the image embeddings generated by the second deep learning image fingerprinting model **1030** will not be compatible with the image embeddings generated by the first deep learning image fingerprinting model **1010**. In some embodiments, the first deep learning image fingerprinting model **1010** and the second deep learning image fingerprinting model **1030** are not different versions of the same model but are completely different image fingerprinting models. In either case, since the resulting image embeddings are not compatible, old image embeddings generated by the first deep learning image fingerprinting model **1010** will need to be converted to be compatible or replaced with image embeddings generated by the second deep learning image fingerprinting model **1030**. Conversion may be the only option if the original images are not available for generating new image embeddings. Both the first deep learning image fingerprinting model **1010** and the second deep learning image fingerprinting model **1030** may be fingerprinting models as described with respect to FIG. **1** in which they generate semantically rich, multi-dimensional feature vectors representing the visual aspects of the input image.

[0093] To train converter model **1020**, a set of training images **1005** may be obtained from training data store **154**. Training images **1005** may be different images than those used to train the first deep learning image fingerprinting model **1010** and different than those used to train the second deep learning image fingerprinting model **1030**. Training images **1005** are input into the first deep learning image fingerprinting model **1010** to generate V1 image embeddings **1015**. Training images **1005** are also input into the second deep learning image fingerprinting model **1030** to generate V2 image embeddings **1035**. V1 image embeddings **1015** are input into converter model **1020** to generate converted image embeddings **1025**. The converted image embeddings **1025** are compared with V2 image embeddings **1035** to calculate a loss using compare and loss calculation **1040**. The loss is backpropagated into converter model **1020** to train converter model **1020**. When training is complete, converter model **1020** will generate converted image embeddings **1025** that are compatible with V2 image embeddings **1035**. The comparison may use Mean Square Error to calculate the loss.

[0094] FIG. **11** illustrates a data flow **1100** for converting a network security system such as

network security system **125** from using the first deep learning image fingerprinting model **1010** to using the second deep learning image fingerprinting model **1030** after converter model **1020** is trained.

[0095] In prior use, image set A **1105** was used to generate V1 image embeddings **1115** stored in embeddings data store **150**. Once second deep learning image fingerprinting model **1030** is ready to use, embedding converter **142** access the V1 image embeddings **1115** and inputs them into converter model **1020** to generate V2 converted image embeddings **1125** and store them in embeddings data store **1130**. Any new images are in image set B **1110** which are fingerprinted by second deep learning image fingerprinting model **1030** to generate V2 original image embeddings **1120**. V2 original image embeddings **1120** and V2 converted image embeddings are stored in embeddings data store **1130**, which may be separate from or the same as embeddings data store **150**.

[0096] Sets of image embeddings from V2 original image embeddings **1120** and V2 converted image embeddings are input into index and ANN model generator **132** to generate various ANN models **1135** and associated ANN indexes **1140**. Because the V2 converted image embeddings **1125** are compatible with V2 original image embeddings **1120**, these generated ANN indexes **1140** and associated ANN models **1135** will accurately identify the most similar image embedding in the relevant ANN index **1140** when prompted with a subject image embedding. Accordingly, new V2 original image embeddings **1120** generated from a subject image can be used by image matcher **134** and image classifier **136** to access relevant ANN models **1135** for matching and classification, respectively.

[0097] FIG. **12** illustrates a method **1200** for training a converter model (e.g., converter model **1020**. Method **1200** begins at **1205** with generating first image fingerprint embeddings for a set of images using the first image fingerprinting model. For example, training images **1005** are provided to converter trainer **144**, and converter trainer **144** inputs the training images **1005** into the first deep learning image fingerprinting model **1010** to generate V1 image embeddings **1015**.

[0098] At **1210**, method **1200** continues with generating second image fingerprint embeddings for the set of images using the second image fingerprinting model. For example, converter trainer **144** inputs the training images **1005** into the second deep learning image fingerprinting model **1030** to generate V2 image embeddings **1035**.

[0099] At **1215** method **1200** continues with providing a first image fingerprint embedding associated with an image of the set of images as input to the converter model. For example, converter trainer **144** takes a first of V1 image embeddings **1015** and inputs it into converter model **1020**.

[0100] At **1220** method **1200** continues with receiving a converted image fingerprint embedding from the converter model. For example, converter model **1020** generates a converted image embedding **1025** in response to the input V1 image embedding **1015**. Each of the V1 image embeddings **1015** are analyzed individually.

[0101] At **1225** method **1200** continues with comparing the converted output image fingerprint embedding with the second image fingerprint embedding associated with the image to calculate a loss. For example, compare and loss calculation **1040** compares the converted image embedding **1025** with the V2 image embedding **1035** that corresponds to the same training image **1005**. Compare and loss calculation **1040** calculates the loss. For example, the comparison may use Mean Square Error to calculate the loss.

[0102] At **1230** method **1200** continues with backpropagating the loss through the converter model. For example, compare and loss calculation **1040** may backpropagate the calculated loss to converter model **1020**.

[0103] At **1235**, method **1200** continues with determining whether there are more images in the set of images. If so, the next image is analyzed by repeating steps **1215**, **1220**, **1225**, and **1235** (e.g., by converting the image embedding with the converter model, calculating the loss, and

backpropagating the loss through the converter model). The repetition continues until no more images are in the set of images and training is complete at **1240**.

[0104] The steps of method **1200** may be performed in a different order than that described including some steps being performed in parallel, in some embodiments. Further, method **1200** may include more or fewer steps than those described.

[0105] FIG. **13** illustrates an exemplary graphical user interface **1300**. Graphical user interface **1300** may be provided via user interface **146** to allow a user to train a new classifier. Other graphical user interfaces may be used to allow a user to configure settings such as setting the threshold values used for matching, configuring security policies, and the like.

[0106] Graphical user interface **1300** provides text box **1305** for a user to enter a classifier name (i.e., a user-defined label). The user can set the training files to positive training data using radio option **1310**. The user can set the training files to negative training data using radio option **1315**. Negative training data may be assigned a default class name and used to further train the classifier. Negative training data is used to add to an index as described with respect to FIG. **6**, but the newly added image embeddings will be given the default label with other negative training data.

[0107] The user may drag and drop or otherwise indicate a location of the image files using entry box **1320**. As few as twenty (20) image files may be provided to train a classifier with the user-defined label (i.e., classifier name).

[0108] The user may use slider **1325** to set a percent match, which is used as the threshold value by labeling **720** and labeling **835**. The default may be any value, though depicted as eighty-five percent (85%). Once the information is entered and submitted using submit button **1330**, classifier trainer **138** can use the information to train the classifier, for example as described with respect to data flow **600** of FIG. **6**.

[0109] FIG. **14** illustrates a computing device **1400**. The computing device **1400** includes various components not included for ease of description in other computing devices discussed herein including, for example, endpoints **105**, network security system **125**, and hosted services **120**. Accordingly, computing device **1400** may be endpoints **105**, network security system **125**, or hosted services **120** by incorporating the functionality described in each.

[0110] Computing device **1400** is suitable for implementing processing operations described herein related to security enforcement, image fingerprinting, image matching, image classification, and image embedding conversion, with which aspects of the present disclosure may be practiced. Computing device **1400** may be configured to implement processing operations of any component described herein including the user system components (e.g., endpoints **105** of FIG. **1**). As such, computing device **1400** may be configured as a specific purpose computing device that executes specific processing operations to solve the technical problems described herein including those pertaining to security enforcement, image fingerprinting, image matching, image classification, and image embedding conversion. Computing device **1400** may be implemented as a single apparatus, system, or device or may be implemented in a distributed manner as multiple apparatuses, systems, or devices. For example, computing device **1400** may comprise one or more computing devices that execute processing for applications and/or services over a distributed network to enable execution of processing operations described herein over one or more applications or services. Computing device **1400** may comprise a collection of devices executing processing for front-end applications/services, back-end applications/services, or a combination thereof. Computing device **1400** includes, but is not limited to, a bus **1405** communicably coupling processors **1410**, output devices **1415**, communication interfaces **1420**, input devices **1425**, power supply **1430**, and memory **1435**.

[0111] Non-limiting examples of computing device **1400** include smart phones, laptops, tablets, PDAs, desktop computers, servers, blade servers, cloud servers, smart computing devices including television devices and wearable computing devices including VR devices and AR devices, e-reader devices, gaming consoles and conferencing systems, among other non-limiting examples.

[0112] Processors **1410** may include general processors, specialized processors such as graphical processing units (GPUs) and digital signal processors (DSPs), or a combination. Processors **1410** may load and execute software **1440** from memory **1435**. Software **1440** may include one or more software components such as image fingerprinter **130**, index and ANN model generator **132**, image matcher **134**, image classifier **136**, classifier trainer **138**, security policy enforcer **140**, embedding converter **142**, converter trainer **144**, user interface **146**, endpoint routing client **110**, or any combination including other software components. In some examples, computing device **1400** may be connected to other computing devices (e.g., display device, audio devices, servers, mobile/remote devices, VR devices, AR devices, etc.) to further enable processing operations to be executed. When executed by processors **1410**, software **1440** directs processors **1410** to operate as described herein for at least the various processes, operational scenarios, and sequences discussed in the foregoing implementations. Computing device **1400** may optionally include additional devices, features, or functionality not discussed for purposes of brevity. For example, software **1440** may include an operating system that is executed on computing device **1400**. Computing device **1400** may further be utilized as endpoints **105** or any of the cloud computing systems in system **100** (FIG. **1**) including network security system **125** or may execute the method **400** of FIG. **4**, method **900** of FIG. **9**, method **1200** of FIG. **12**, or any combination.

[0113] Referring still to FIG. **14**, processors **1410** may include a processor or microprocessor and other circuitry that retrieves and executes software **1440** from memory **1435**. Processors **1410** may be implemented within a single processing device but may also be distributed across multiple processing devices or sub-systems that cooperate in executing program instructions. Examples of processors **1410** include general purpose central processing units, microprocessors, graphical processing units, application specific processors, sound cards, speakers and logic devices, gaming devices, VR devices, AR devices as well as any other type of processing devices, combinations, or variations thereof.

[0114] Memory **1435** may include any computer-readable storage device readable by processors **1410** and capable of storing software **1440** and data stores **1445**. Data stores **1445** may include model data store **148**, embedding data store **150**, security policy data store **152**, training data store **154**, and index data store **156**. Memory **1435** may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, cache memory, or other data. Examples of storage media include random access memory, read only memory, magnetic disks, optical disks, flash memory, virtual memory and non-virtual memory, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or other suitable storage media, except for propagated signals. In no case is the computer-readable storage device a propagated signal.

[0115] In addition to computer-readable storage devices, in some implementations, memory **1435** may also include computer-readable communication media over which at least some of software **1440** may be communicated internally or externally. Memory **1435** may be implemented as a single storage device but may also be implemented across multiple storage devices or sub-systems co-located or distributed relative to each other. Memory **1435** may include additional elements, such as a controller, capable of communicating with processors **1410** or possibly other systems.

[0116] Software **1440** may be implemented in program instructions and among other functions may, when executed by processors **1410**, direct processors **1410** to operate as described with respect to the various operational scenarios, sequences, and processes illustrated herein. For example, software **1440** may include program instructions for executing image fingerprinting, image matching, image classifying, image embedding conversion, or security policy enforcement as described herein.

[0117] In particular, the program instructions may include various components or modules that cooperate or otherwise interact to conduct the various processes and operational scenarios

described herein. The various components or modules may be embodied in compiled or interpreted instructions, or in some other variation or combination of instructions. The various components or modules may be executed in a synchronous or asynchronous manner, serially or in parallel, in a single threaded environment or multi-threaded, or in accordance with any other suitable execution paradigm, variation, or combination thereof. Software **1440** may include additional processes, programs, or components, such as operating system software, virtual machine software, or other application software. Software **1440** may also include firmware or some other form of machine-readable processing instructions executable by processors **1410**.

[0118] In general, software **1440** may, when loaded into processors **1410** and executed, transform a suitable apparatus, system, or device (of which computing device **1400** is representative) overall from a general-purpose computing system into a special-purpose computing system customized to execute specific processing components described herein as well as process data and respond to queries. Indeed, encoding software **1440** on memory **1435** may transform the physical structure of memory **1435**. The specific transformation of the physical structure may depend on various factors in different implementations of this description. Examples of such factors may include, but are not limited to, the technology used to implement the storage media of memory **1435** and whether the computer-storage media are characterized as primary or secondary storage, as well as other factors.

[0119] For example, if the computer readable storage device are implemented as semiconductor-based memory, software **1440** may transform the physical state of the semiconductor memory when the program instructions are encoded therein, such as by transforming the state of transistors, capacitors, or other discrete circuit elements constituting the semiconductor memory. A similar transformation may occur with respect to magnetic or optical media. Other transformations of physical media are possible without departing from the scope of the present description, with the foregoing examples provided only to facilitate the present discussion.

[0120] Communication interfaces **1420** may include communication connections and devices that allow for communication with other computing systems (not shown) over communication networks (not shown). Communication interfaces **1420** may also be utilized to cover interfacing between processing components described herein. Examples of connections and devices that together allow for inter-system communication may include network interface cards or devices, antennas, satellites, power amplifiers, RF circuitry, transceivers, and other communication circuitry. The connections and devices may communicate over communication media to exchange communications with other computing systems or networks of systems, such as metal, glass, air, or any other suitable communication media. The aforementioned media, connections, and devices are well known and need not be discussed at length here.

[0121] Communication interfaces **1420** may also include associated user interface software executable by processors **1410** in support of the various user input and output devices discussed below. Separately or in conjunction with each other and other hardware and software elements, the user interface software and user interface devices may support a graphical user interface, a natural user interface, or any other type of user interface, for example, that enables front-end processing and including rendering of user interfaces, such as a user interface that is used by a user on endpoint **105**. Exemplary applications/services may further be configured to interface with processing components of computing device **1400** that enable output of other types of signals (e.g., audio output, handwritten input) in conjunction with operation of exemplary applications/services (e.g., a collaborative communication application/service, electronic meeting application/service, etc.) described herein.

[0122] Input devices **1425** may include a keyboard, a mouse, a voice input device, a touch input device for receiving a touch gesture from a user, a motion input device for detecting non-touch gestures and other motions by a user, gaming accessories (e.g., controllers and/or headsets) and other comparable input devices and associated processing elements capable of receiving user input from a user. Output devices **1415** may include a display, speakers, haptic devices, and the like. In

some cases, the input and output devices may be combined in a single device, such as a display capable of displaying images and receiving touch gestures. The aforementioned user input and output devices are well known in the art and need not be discussed at length here.

[0123] Communication between computing device **1400** and other computing systems (not shown), may occur over a communication network or networks and in accordance with various communication protocols, combinations of protocols, or variations thereof. Examples include intranets, internets, the Internet, local area networks, wide area networks, wireless networks, wired networks, virtual networks, software defined networks, data center buses, computing backplanes, or any other type of network, combination of network, or variation thereof. The aforementioned communication networks and protocols are well known and need not be discussed at length here. However, some communication protocols that may be used include, but are not limited to, the Internet protocol (IP, IPv4, IPv6, etc.), the transfer control protocol (TCP), and the user datagram protocol (UDP), as well as any other suitable communication protocol, variation, or combination thereof.

[0124] The computing device **1400** has a power supply **1430**, which may be implemented as one or more batteries. The power supply **1430** may further include an external power source, such as an AC adapter or a powered docking cradle that supplements or recharges the batteries. In some embodiments, the power supply **1430** may not include batteries and the power source may be an external power source such as an AC adapter.

[0125] The following specific embodiments are contemplated from the above description.

Image Matching Embodiments

[0126] A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus (i.e., processors), cause the apparatus to perform the actions. One general aspect includes a computer-implemented method. An image fingerprinter may be used to generate image fingerprint embeddings by providing each of the images as input to an image fingerprinting model trained to generate an image fingerprint embedding that is a multi-dimensional feature vector representing visual aspects of the input image. The image fingerprinter may receive each of the image fingerprint embeddings as output from the image fingerprinting model. An index and approximate nearest neighbors model generator may generate an index of the image fingerprint embeddings and an approximate nearest neighbors model associated with the index. An image matcher may analyze a subject image by providing the subject image as input to the image fingerprinting model and receiving a subject image fingerprint embedding as output from the image fingerprinting model. The image matcher may provide the subject image fingerprint embedding as input to the approximate nearest neighbors model trained to retrieve a most similar image fingerprint embedding from the index and generate a score indicating a similarity of the subject image fingerprint embedding to the most similar image fingerprint embedding. Image matcher may analyze the score received from the approximate nearest neighbors model and a security enforcement module may perform a security action based on the score. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0127] Implementations may include one or more of the following features. In some embodiments, the score indicates a normalized value of an angular distance between the subject image fingerprint embedding and the most similar image fingerprint embedding. In some embodiments, the image matcher may scan a cloud-based data store for subject images and analyze each of the subject images. In some embodiments, analyzing the score may include comparing the score to a threshold value. In some embodiments, the security action may include ignoring the subject image when the

score is below the threshold value, and the security action may include transmitting a security message to an administrator indicating the subject image when the score meets or exceeds the threshold value. In some embodiments, the network security system identifies the subject image based on a user performing an action including the subject image. The action may include uploading the subject image to a cloud application; downloading the subject image from the cloud application; deleting the subject image from the cloud application; sharing the subject image on the cloud application; moving the subject image within the cloud application; and moving the subject image outside the cloud application. In some embodiments, the image matcher may compare the score to a threshold value and the security action may include blocking the action when the score meets or exceeds the threshold value. In some embodiments, the subject image is a transformed copy of the image used to generate the most similar image fingerprint embedding. In some embodiments, the subject image is transformed by cropping the image; rotating the image; distorting color of the image; or blurring the image. In some embodiments, the network security system may provide a graphical user interface including user input elements selectable by a user for providing the images, and the network security system may receive, via the graphical user interface, an indication of the location of the images (e.g., the images may be uploaded to the network security system). Implementations of the described techniques may include hardware, a method or process, or computer software on a computer-accessible medium.

[0128] Another general aspect includes a network security system. The network security system includes one or more processor and one or more computer-readable memory devices. The memory devices may include an image fingerprinting model trained to generate an image fingerprint embedding that is a multi-dimensional feature vector representing visual aspects of an input image. The memory devices may further include an image matching component including instructions that, upon execution by the one or more processors, cause the one or more processors to generate image fingerprint embeddings using the image fingerprinting model. The image matching component may also generate an index of the image fingerprint embeddings and an approximate nearest neighbors model associated with the index. The image matching component may further analyze a subject image by providing the subject image as input to the image fingerprinting model and receiving a subject image fingerprint embedding as output from the image fingerprinting model. The image matching component may further provide the subject image fingerprint embedding as input to the approximate nearest neighbors model trained to retrieve a most similar image fingerprint embedding from the index and generate a score indicating a similarity of the subject image fingerprint embedding to the most similar image fingerprint embedding. The image matching component may further analyze the score received from the approximate nearest neighbors model and perform a security action based on the score. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0129] Implementations may include one or more of the following features. In some embodiments, the score indicates a normalized value of an angular distance between the subject image fingerprint embedding and the most similar image fingerprint embedding. In some embodiments, a cloud-based storage is scanned for subject images and the image matching component may further analyze each of the subject images. In some embodiments, the score is compared to a threshold value. The security action may include ignoring the subject image when the score is below the threshold value, and the security action may include transmitting a security message to an administrator indicating the subject image when the score meets or exceeds the threshold value. In some embodiments, the network security system includes an image receiving component including instructions that, upon execution by the one or more processors, cause the one or more processors to identify the subject image based on a user performing an action including the subject image. The action may include one of uploading the subject image to a cloud application; downloading the subject image from the cloud application; deleting the subject image from the cloud application;

sharing the subject image on the cloud application; moving the subject image within the cloud application; and moving the subject image outside the cloud application. In some embodiments, the score is compared to a threshold value and the security action may include blocking the action when the score meets or exceeds the threshold value. In some embodiments, the subject image is a transformed copy of the image used to generate the most similar image fingerprint embedding. In some embodiments, the subject image is transformed by one or more of cropping the image; rotating the image; distorting color of the image; and blurring the image. In some embodiments, the network security system may include a graphical user interface component used to provide a graphical user interface with user input elements selectable by a user for providing the images. The network security system may receive, via the graphical user interface, an indication of a location of the images. Implementations of the described techniques may include hardware, a method or process, or computer software on a computer-accessible medium.

Image Classifier Training and Image Classification Embodiments

[0130] A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions. One general aspect includes a computer-implemented method. The method may include generating a first plurality of image fingerprint embeddings, the generating may include providing each of a first plurality of images as input to an image fingerprinting model, where the image fingerprinting model is trained to generate an image fingerprint embedding that is a multi-dimensional feature vector representing visual aspects of the input image, and receiving each of the first plurality of image fingerprint embeddings as output from the image fingerprinting model. The method also includes generating a first index of the first plurality of image fingerprint embeddings and a first approximate nearest neighbors model associated with the first index used to classify subject images, where generating the first index may include clustering subsets of the first plurality of image fingerprint embeddings into embedding clusters. The method also includes applying a label to each of the embedding clusters. The method further includes receiving a second plurality of images selected by a user and a user-defined label for the second plurality of images and generating a second plurality of image fingerprint embeddings using the image fingerprinting model. The method also includes adding the second plurality of image fingerprint embeddings to the first index to generate a second index and a second approximate nearest neighbors model associated with the second index, the adding may include: clustering the second plurality of image fingerprint embeddings into a user embedding cluster, and applying the user-defined label to the user embedding cluster. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0131] Implementations may include one or more of the following features. The computer-implemented method where the second plurality of images may include between twenty (20) and one hundred (100) images. The computer-implemented method may include analyzing a subject image, the analyzing may include: providing the subject image as input to the image fingerprinting model; receiving a subject image fingerprint embedding as output from the image fingerprinting model; providing the subject image fingerprint embedding as input to the second approximate nearest neighbors model trained to retrieve a label of a most similar image fingerprint embedding from the second index and generate a score indicating a similarity of the subject image fingerprint embedding to the most similar image fingerprint embedding; analyzing the score received from the second approximate nearest neighbors model; and labeling the subject image based on the score. In some embodiments, the computer-implemented method may include: applying a security policy to the subject image based on the label. In some embodiments, analyzing the score may include

comparing the score to a threshold value, labeling the subject image may include labeling the subject image with the label of the most similar image fingerprint embedding when the score meets or exceeds the threshold value, and labeling the subject image may include labeling the subject image with a default label when the score is below the threshold value. In some embodiments, the score indicates a normalized value of an angular distance between the subject image fingerprint embedding and the most similar image fingerprint embedding. In some embodiments, the computer-implemented method may include: identifying the subject image based on a user performing an action including the subject image. The action may include one of: uploading the subject image to a cloud application; downloading the subject image from the cloud application; deleting the subject image from the cloud application; sharing the subject image on the cloud application; moving the subject image within the cloud application; and moving the subject image outside the cloud application. In some embodiments, the computer-implemented method may include: generating a plurality of negative image fingerprint embeddings, the generating may include: providing each of a plurality of negative images as input to the image fingerprinting model, and receiving each of the plurality of negative image fingerprint embeddings as output from the image fingerprinting model; and adding the plurality of negative image fingerprint embeddings to one or more of the first index and the second index, the adding may include labeling each negative image fingerprint embedding with a default label. In some embodiments, the computer-implemented method may include: providing a graphical user interface having user input elements selectable by a user for providing the second plurality of images and the user-defined label. In some embodiments, the computer-implemented method may include: receiving a third plurality of images selected by the user and a second user-defined label for the third plurality of images; generating a third plurality of image fingerprint embeddings, the generating may include: providing each of the third plurality of images as input to the image fingerprinting model, and receiving each of the third plurality of image fingerprint embeddings as output from the image fingerprinting model; and adding the third plurality of image fingerprint embeddings to the first index to generate a third index and a third approximate nearest neighbors model associated with the third index, the adding may include: clustering the third plurality of image fingerprint embeddings into a user embedding cluster, and applying the second user-defined label to the user embedding cluster. In some embodiments, the computer-implemented method may include analyzing a subject image, the analyzing may include: providing the subject image as input to the image fingerprinting model; receiving a subject image fingerprint embedding as output from the image fingerprinting model; providing the subject image fingerprint embedding as input to the second approximate nearest neighbors model trained to retrieve a first label of a most similar image fingerprint embedding from the second index and generate a first score indicating a similarity of the subject image fingerprint embedding to the most similar image fingerprint embedding from the second index; providing the subject image fingerprint embedding as input to the third approximate nearest neighbors model trained to retrieve a second label of a most similar image fingerprint embedding from the third index and generate a second score indicating a similarity of the image subject fingerprint embedding to the most similar image fingerprint embedding from the third index; analyzing the first score and the second score; and labeling the subject image based on the first score and the second score. In some embodiments, the computer-implemented method may include: based on a determination that the first score is higher than the second score, labeling the subject image with the first label. In some embodiments, the computer-implemented method may include: based on a determination that the first score and the second score each meet or exceed a threshold value, labeling the subject image with the first label and the second label. In some embodiments, the computer-implemented method may include: based on a determination that the first score and the second score are below a threshold value, labeling the subject image with a default label. Implementations of the described techniques may include hardware, a method or process, or computer software on a computer-accessible medium. Other aspects may include a network

security system implementing the computer-implemented method.

Image Embedding Conversion and Embedding Converter Training Embodiments

[0132] A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions. One general aspect includes a computer-implemented method including generating a plurality of first image fingerprint embeddings, the generating may include: providing each of a plurality of images as input to a first image fingerprinting model, where the first image fingerprinting model is trained to generate a first image fingerprint embedding that is a multi-dimensional feature vector representing visual aspects of the input image, and receiving each of the plurality of first image fingerprint embeddings as output from the first image fingerprinting model. The method also includes generating a plurality of second image fingerprint embeddings, the generating may include: providing each of the plurality of images as input to a second image fingerprinting model, where the second image fingerprinting model is trained to generate a second image fingerprint embedding that is a multi-dimensional feature vector representing visual aspects of the input image, and receiving each of the plurality of second image fingerprint embeddings as output from the second image fingerprinting model, where the first image fingerprint embedding is different than the second image fingerprint embedding generated for each of the plurality of images. The method also includes training a converter model to convert a first image fingerprint embedding to a second image fingerprint embedding compatible with other second image fingerprint embeddings, the training may include: for each image of the plurality of images: providing the first image fingerprint embedding associated with the respective image as input to the converter model, receiving an output image fingerprint embedding from the converter model, comparing the output image fingerprint embedding with the second image fingerprint embedding associated with the respective image to calculate a loss, and backpropagating the loss through the converter model. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0133] Implementations may include one or more of the following features. In some embodiments, the converter model is an autoencoder. In some embodiments, the comparing uses mean square error to calculate the loss. In some embodiments, the first image fingerprinting model is a first version of an image fingerprinting model, and the second image fingerprinting model is a second version of the image fingerprinting model. In some embodiments, a conversion second image fingerprint embedding generated by the converter model and an original second image fingerprint embedding generated by the second image fingerprinting model are not identical but are exchangeable. In some embodiments, the plurality of images may include images that were not used to train the first image fingerprinting model or the second image fingerprinting model. In some embodiments, the method further includes accessing a plurality of stored first image fingerprint embeddings, providing each of the stored first image fingerprint embeddings as input to the converter model to generate a second plurality of second image fingerprint embeddings, and adding the second plurality of second image fingerprint embeddings to an index including a third plurality of second image fingerprint embeddings generated by the second image fingerprinting model, where the index is used by an approximate nearest neighbors model. In some embodiments, the computer-implemented method may include analyzing a subject image, the analyzing may include: providing the subject image as input to the second image fingerprinting model; receiving a subject image fingerprint embedding as output from the second image fingerprinting model; providing the subject image fingerprint embedding as input to the approximate nearest neighbors model trained to retrieve a most similar image fingerprint embedding from the index and generate a

score indicating a similarity of the subject image fingerprint embedding to the most similar image fingerprint embedding; analyzing the score received from the approximate nearest neighbors model; and performing a security action based on the score. In some embodiments, the computer-implemented method may include: clustering subsets of the second image fingerprint embeddings in the index into embedding clusters; and applying a label to each of the embedding clusters. In some embodiments, the computer-implemented method may include analyzing a subject image, the analyzing may include: providing the subject image as input to the second image fingerprinting model; receiving a subject image fingerprint embedding as output from the second image fingerprinting model; providing the subject image fingerprint embedding as input to the approximate nearest neighbors model trained to retrieve a label of a most similar second image fingerprint embedding from the index and generate a score indicating a similarity of the subject image fingerprint embedding to the most similar second image fingerprint embedding; analyzing the score received from the approximate nearest neighbors model; and labeling the subject image based on the score. Implementations of the described techniques may include hardware, a method or process, or computer software on a computer-accessible medium. Other aspects may include a network security system implementing the computer-implemented method.

[0134] The aforementioned discussion is presented to enable any person skilled in the art to make and use the technology disclosed and is provided in the context of a particular application and its requirements. Various modifications to the disclosed implementations will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other implementations and applications without departing from the spirit and scope of the technology disclosed. Thus, the technology disclosed is not intended to be limited to the implementations shown but is to be accorded the widest scope consistent with the principles and features disclosed herein.

[0135] Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense, as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to." As used herein, the terms "connected," "coupled," or any variant thereof means any connection or coupling, either direct or indirect, between two or more elements; the coupling or connection between the elements can be physical, logical, or a combination thereof. Additionally, the words "herein," "above," "below," and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. Where the context permits, words in the above Detailed Description using the singular or plural number may also include the plural or singular number, respectively. The word "or," in reference to a list of two or more items, covers all of the following interpretations of the word: any of the items in the list, all of the items in the list, and any combination of the items in the list.

[0136] The phrases "in some embodiments," "according to some embodiments," "in the embodiments shown," "in other embodiments," and the like generally mean the particular feature, structure, or characteristic following the phrase is included in at least one implementation of the present technology and may be included in more than one implementation. In addition, such phrases do not necessarily refer to the same embodiments or different embodiments.

[0137] The above Detailed Description of examples of the technology is not intended to be exhaustive or to limit the technology to the precise form disclosed above. While specific examples for the technology are described above for illustrative purposes, various equivalent modifications are possible within the scope of the technology, as those skilled in the relevant art will recognize. For example, while processes or blocks are presented in a given order, alternative implementations may perform routines having steps, or employ systems having blocks, in a different order, and some processes or blocks may be deleted, moved, added, subdivided, combined, and/or modified to provide alternative or subcombinations. Each of these processes or blocks may be implemented in a variety of different ways. Also, while processes or blocks are at times shown as being performed in

series, these processes or blocks may instead be performed or implemented in parallel or may be performed at different times. Further any specific numbers noted herein are only examples: alternative implementations may employ differing values or ranges.

Claims

1. A network security system, comprising: an image embedding model trained to receive a subject image and generate an image embedding representing visual aspects of the subject image; an embedding datastore comprising a plurality of image embeddings generated by the image embedding model; a classifier training component configured to: receive a user input comprising a user-defined label and a set of user-submitted images, wherein the set of user-submitted images are different than images used to generate the plurality of image embeddings, use the image embedding model to generate training image embeddings of the images in the set of user-submitted images, and in response to receiving the user input, train a user-defined image classifier based on the user input, wherein the user-defined image classifier comprises an index and an approximate nearest neighbors model, wherein to train the user-defined image classifier, the classifier training component is configured to: generate the index comprising at least a subset of the plurality of image embeddings from the embedding datastore and the training image embeddings; generate a user-defined embedding cluster of the training image embeddings; apply the user-defined label to the user-defined embedding cluster; and generate the approximate nearest neighbors model, wherein the approximate nearest neighbors model is trained to classify the subject images using the index and the image embedding generated by the image embedding model based on the subject image.
2. The network security system of claim 1, wherein the set of user-submitted images contains between twenty (20) and one hundred (100) images.
3. The network security system of claim 1, further comprising: the trained user-defined image classifier configured to classify a new subject image, wherein to classify the new subject image, the trained user-defined image classifier is configured to: use the image embedding model to generate a new subject image embedding; use the approximate nearest neighbors model and the index to retrieve a label of a most similar image embedding from the index and generate a score indicating a similarity of the new subject image embedding to the most similar image embedding; analyze the score; and label the new subject image based on the score.
4. The network security system of claim 3, further comprising: a security policy enforcement component configured to apply a security policy to the new subject image based at least in part on the label.
5. The network security system of claim 3, wherein to analyze the score and label the new subject image based on the score, the trained user-defined image classifier is further configured to: compare the score to a threshold value; label the new subject image with the label of the most similar image embedding when the score equals or exceeds the threshold value; and label the new subject image with a default label when the score is below the threshold value.
6. The network security system of claim 3, wherein the score comprises a value of zero to one hundred (0-100) that indicates an angular distance between the new subject image embedding and the most similar image embedding.
7. The network security system of claim 3, further comprising: a security policy enforcement component configured to identify the new subject image based on a user performing an action including the new subject image.
8. The network security system of claim 7, wherein the action comprises one of: uploading the subject image to a cloud application; downloading the subject image from the cloud application; deleting the subject image from the cloud application; sharing the subject image on the cloud application; moving the subject image within the cloud application; and moving the subject image

outside the cloud application.

9. The network security system of claim 1, wherein the classifier training component is further configured to: use the image embedding model to generate a plurality of negative image embeddings, wherein the subset of the plurality of image embeddings comprises the plurality of negative image embeddings; and label each negative image embedding with a default label in the index.

10. The network security system of claim 1, further comprising: a user interface component configured to: provide a graphical user interface comprising selectable user input elements usable to upload the set of user-submitted images and the user-defined label.

11. The network security system of claim 1, wherein the classifier training component is further configured to: receive a second user input comprising a second set of user-submitted images and a second user-defined label; use the image embedding model to generate second training image embeddings of the images in the second set of user-submitted images; in response to receiving the second user input, train a second user-defined image classifier based on the second user input, wherein the second user-defined image classifier comprises a second index and a second approximate nearest neighbors model, wherein to train the second user-defined image classifier, the classifier training component is configured to: generate the second index comprising the at least the subset of the plurality of image embeddings from the embedding datastore and the second training image embeddings; generate a second user-defined embedding cluster of the second training image embeddings; apply the second user-defined label to the second user-defined embedding cluster; and generate the second approximate nearest neighbors model, wherein the second approximate nearest neighbors model is trained to classify the subject image using the second index and the image embedding generated by the image embedding model based on the subject image.

12. The network security system of claim 11, further comprising: a trained multi-label image classifier configured to: receive a new subject image; use the image embedding model to generate a new subject embedding; use the user-defined image classifier to generate a first label and a first score; use the second user-defined image classifier to generate a second label and a second score; analyze the first score and the second score; and label the new subject image based on the first score and the second score.

13. The network security system of claim 12, wherein the trained multi-label image classifier is further configured to: based on a determination that the first score is higher than the second score, label the subject image with the first label.

14. The network security system of claim 12, wherein the trained multi-label image classifier is further configured to: based on a determination that the first score and the second score exceed a threshold value, label the subject image with the first label and the second label.

15. The network security system of claim 12, wherein the trained multi-label image classifier is further configured to: based on a determination that the first score and the second score are below a threshold value, label the subject image with a default label.

16. A computer-implemented method, comprising: generating initial image embeddings using an image embedding model, wherein the initial image embeddings represent visual aspects of a corresponding image; receiving a user input comprising a user-defined label and a set of user-submitted images, wherein the user-submitted images in the set of user-submitted images are different from the corresponding images used to generate the initial image embeddings; generating training image embeddings for the set of user-submitted images with the image embedding model; in response to receiving the user input, training a user-defined image classifier based on the user input, wherein the user-defined image classifier comprises an index and an approximate nearest neighbors model, the training comprising: generating the index comprising at least a subset of the initial image embeddings and the training image embeddings; generating a user-defined embedding cluster of the training image embeddings; applying the user-defined label to the user-defined embedding cluster; and generating the approximate nearest neighbors model, wherein the

approximate nearest neighbors model is trained to classify a subject image using the index and a subject image embedding generated by the image embedding model based on the subject image.

17. The computer-implemented method of claim 16, wherein the set of user-submitted images contains between twenty (20) and one hundred (100) images.

18. The computer-implemented method of claim 16, further comprising: receiving a new subject image; generating a new subject image embedding of the new subject image; submitting the new subject image embedding to the approximate nearest neighbors model to retrieve a label of a most similar image embedding from the index and a score indicating a similarity of the new subject image embedding to the most similar subject image embedding; analyzing the score; and labeling the new subject image based on the score.

19. The computer-implemented method of claim 18, further comprising: applying a security policy to the new subject image based at least in part on the label.

20. The computer-implemented method of claim 18, wherein the analyzing the score and the labeling the new subject image based on the score comprises: comparing the score to a threshold value; based on a determination that the score equals or exceeds the threshold value, labeling the new subject image with the label of the most similar image embedding; and based on a determination that the score is below the threshold value, labeling the new subject image with a default label.