Content-Based Retrieval of Mars Images

David Hoffmann, Niklas Sprengel, and Kim Schwarz

***Abstract*—In recent years the volume of remote sensing databases has increased significantly which raises the need for systems that help the user to efficiently analyse and use this data. Such systems are content-based remote sensing image retrieval (CBIR) systems, which retrieve images similar to a given query image based on their semantic content. Conventional CBIR approaches rely on hand-crafted features, which lack the ability to represent high-level semantic content and are not optimised for retrieval. More recent approaches have shown that better feature representations can be generated by employing Deep Neural Networks (DNN). However training DNNs is often time consuming and usually requires a significant amount of training data. In this work we introduce a dedicated Mars Image Retrieval System (MIRS) to tackle the challenges of CBIR for Mars remote sensing data sets. MIRS utilises pretrained deep neural networks (DNN) to encode images into meaningful high-level feature representations. The representations are then embedded into a similarity preserving hashing space by a hashing network, which allows for scalable image storage and retrieval. This greatly reduces the required training time and allows scalability to large data sets. Furthermore, MIRS can be trained in a supervised as well as an unsupervised way to allow flexibility as labelled data sets of Mars images are scarce while unlabelled data is widely available. To ensure the adaptability of MIRS to unknown data sets we additionally include two approaches to improve its generalisation capability and allow it to be used for different domains without requiring retraining. Comparisons with a standard feature based DNN approach have shown that the generated hash codes provided an improvement in both retrieval accuracy and efficiency.**

I. Introduction

UNDERSTANDING the surface conditions and geological properties of a planetary body are essential for mission planning. Hence, maps that accurately outline geological features and allow a geomorphic and stratigraphic analysis of the planetary body can be of great importance. Planetary maps are most commonly produced by teams of experts, including cartographers, artists and scientists, which is a costly and time consuming process.

With the rise of massive remote sensing (RS) image archives, methods were developed that automatically map geological features. This also led to the emergence of content-based remote sensing image retrieval (CBIR) systems that allow even users with little domain expertise to draw important information from the data. This is important because manual labelling of data sets by experts is expensive and impractical for very large data sets. Rather than producing geological maps, these systems retrieve images similar to a given query image based on their content. Thus, they carry great potential to improve automatic planetary surface analysis while decreasing the required expertise and costs of producing planetary maps [1]. A basic CBIR system is shown in Figure 1. It consist of an image characterisation step, where the semantic and spatial information of the images are encoded into meaningful features, and an image retrieval step where the most similar images to a query are returned from a database. [2].

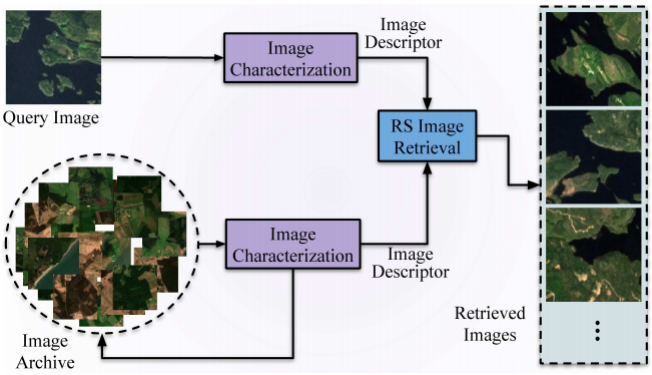

Fig. 1. Pipeline of a general CBIR system [2].

For the image characterisation step, traditional CBIR systems rely on handcrafted features [3]–[5], which are limited in their capability to represent high-level semantic content and are not optimised for the retrieval task [6]. After obtaining the feature representations the query image has to be compared with all images in the archive to retrieve the most similar images, which is sometimes done by calculating the Euclidean distances or by applying a k-nearest neighbour (kNN) algorithm [7]. This, however, is very inefficient and time consuming for large image databases [2].

To tackle these challenges we introduce MIRS, a CBIR system for Mars images that combines recent advances in the field of supervised and unsupervised deep neural networks (DNN) with hashing to tackle some of the issues described above. DNNs have led to a significant performance increase in image retrieval [2]. They consist of multiple layers which can learn representations of data with varying levels of complexity. Thus, they are able to automatically extract high-level features which outperform those created manually [8]. To ensure the scalability of our approach we designed MIRS to generate similarity preserving hash codes. Hashing based retrieval methods learn to map high-dimensional features to low-dimensional binary code while preserving similarity. The hash code representations of the images occupy considerably less storage and retrieval speed is greatly increased [9] [6]. Because we aim to develop a highly flexible and adaptive system, MIRS allows for both a supervised and an unsupervised learning approach with additional generalisation capability injection. This is important because a supervised system can only be applied to annotated data sets, but the amount of unlabelled data exceeds the amount of annotated data by far, limiting the possible application of supervised systems. MIRS circumvents this drawback by

being able to adjust to unsupervised learning and therefore does not rely on data labels. An additional challenge for most CBIR systems is that satellite images are subjected to changes in lightning, angles and ground conditions. As a consequence, networks trained on data sets that are obtained from a single domain, struggle to adapt well to data sampled from a different domain. By integrating domain adaptation we ensure strong generalisation capabilities of our system and high performance on unseen data [10].

The rest of this paper is organised as follows: In Section II we present work related to this paper. Next, we introduce important background and present the theoretical concepts of our model (Section III). In Section IV we first describe the details of our experimental setup and then dive into the sensitivity analysis, where we thoroughly evaluate the results for different configurations of hyper-parameters and training approaches in terms of precision and retrieval accuracy. In Section IV-C and IV-D we compare the results of our best MIRS configurations with each other and the baseline respectively. Finally, we also discuss the training and inference speed in Section IV-E. At last, we summarise our findings and draw our conclusions in the final section (Section V).

II. Related Work

An ever-growing amount of cameras provides a constant stream of new image data which requires further analysis. Thus, developing systems that allow the user to efficiently access and use the acquired data becomes increasingly relevant. This is especially true for Remote Sensing Image Archives. Improvements in the capturing capabilities of satellite cameras as well as a significant reduction in the launch costs to deploy such systems have led to an increase in available earth observation data composing large databases. These archives require dedicated CBIR systems to extract specific images and supply them to automated analysis software to infer relevant information at scale.

Early CBIR systems were often based on classical computer vision approaches which relied on a set of low-level hand-crafted feature representations for a given image such as basic texture and colour features [11], [12], shape features [12], [13], spatial features [14] and more advanced techniques like Generic Fourier Descriptors [15] among many others. After feature encoding, similar images could then be identified by directly comparing these features using a distance metric like the Euclidean distance.

A key problem of these approaches is that they still required significant processing of the generated features and therefore could not handle large scale databases. To deal with this problem some approaches further enhanced the basic feature comparison by using the extracted features to compute binary hash codes which could then be used for faster and more efficient similarity comparisons [16], [17]. While all these approaches provided functional CBIR systems they still relied on a lot of human fine-tuning and were often lacking in terms of accuracy and retrieval performance.

More modern CBIR systems addressed this issue by replacing the hand-crafted feature generation with DNNs, which have, in recent years, delivered breakthrough performances in various image related tasks. Deep learning based hashing approaches began to use DNNs to directly generate feature representations that can easily be transformed into binary hash codes. As a consequence, the performance of CBIR systems drastically increased.

One early system to utilise this concept was introduced by Zhu et al. in [18]. The authors based their design on an established deep neural network architecture and modified the last layer of the model to output a K-dimensional encoding with K corresponding to the final hash length.

The model was then trained in a supervised manner using pairwise similarity labels and a loss function consisting of the pairwise cross-entropy loss and a special quantization loss which pushes the output values of the last layer to values close to -1 or 1. The final hashes were then computed by applying a sign function to the last feature output.

This general architecture design has been utilised by many other following research projects, many of which were reviewed by Li et al. in [19]. Based on this they proposed a combined Deep Hashing Neural Network (DHNN) architecture including multiple concepts from its predecessors to be used on a large scale remote sensing CBIR task. Their design also utilises the cross-entropy loss for training the separation of dissimilar inputs. Additionaly they introduced two different quantization losses based on $L1$ and $L2$ norm regularisation respectively and evaluated their respective performance for the CBIR task.

The authors of [20] further improved the general design of DHNNs in RS by introducing the semantic class information into the training procedure. This was achieved by adding a fully connected classification layer on top of the hashing network. The classification layer was followed by a softmax function and then optimised with standard cross-entropy loss. By combining the loss for the hashing component with the cross-entropy loss their design could better discriminate between the various classes and delivered hashing based image retrieval and classification results at the same time.

Therefore, many other approaches in deep metric learning rely on triplet loss to convey the semantic information to simultaneously consider similarity information between images and their respective class association. Triplet loss computes the similarity between images based on their associated class labels and a distance metric computed over the feature outputs to specifically select fitting samples for the current training step. In [21] triplet loss is directly used to train a neural network for CBIR on RS images by comparing the Euclidean distance of the output features.

Triplet loss was also utilised by Roy et al. [6] who proposed a Metric-Learning based Deep Hashing Network (MiLaN). MiLaN is a dedicated hashing network trained on the output of a deep feature extractor pre-trained on ImageNet. This allows the output of the feature extractor to consist of generalising features while enabling the hashing component to generate the hash codes based on mainly high-level features. Their design will be described in further detail in III as our approach is

primarily based on their architecture.
Another relevant field of research is focused on ensuring that deep hashing networks can accurately compute hashes for input images from different imaging sources other than just the ones available in the training data set. The authors of [22] for example have mainly focused at a multitude of image transformations which can be applied to a set of multi-spectral training images to make the resulting model robust to spectral changes in the input images.
A similar problem has been tackled by Li et al. [23] who define separate hashing networks for panchromatic data with a high spatial resolution and multispectral data with a lower spatial resolution. These two designs are then jointly optimised in the training process to ensure that the feature output of one of the networks corresponds to a similar feature output in the other network. Hence, they are able to perform image retrieval with panchromatic input images on multispectral images and the other way around.
Other approaches have attempted to incorporate the spatial information within input images into the training for ranking outputs. In this context, [24] proposes a novel hierarchical Recurrent Neural Network (RNN) architecture which processes the output from hierarchically structured Convolutional Neural Networks (CNN) feature maps to compute the hash codes. This promises to simultaneously analyse the spatial information represented by the different CNN layers as well as the semantic information which is still being selected for by a ranking loss function.
A radically different approach is explored in [25] where the authors describe a process to not only quantise feature outputs of a model but also its weights in the feature extraction layers to achieve a smaller memory footprint and faster processing speeds.
Testing this approach on various deep hashing neural networks, they achieve a significant increase in processing speed with only minor performance degradation. This is intended to be used in environments with low processing capabilities and power constraints like satellite systems and could potentially be used to integrate the hash code generation directly into the acquisition step of building remote sensing image archives.
All of the aforementioned approaches are supervised designs and therefore rely on labelled training data. On the other hand, some authors have proposed unsupervised training methods for DHNNs. For example, in [26] the authors generated pseudo-labels by using a K-means clustering algorithm and trained a deep neural network for hashing using an elaborate optimisation process. While such unsupervised approaches are beneficial due to reducing training data requirements they still lack behind more established supervised designs in terms of performance.

III. Methodology

A. Mars Image Retrieval System (MIRS)

MIRS, the content-based image retrieval system we built, uses a deep learning network as a feature extractor. These feature vectors are then transformed into binary hash codes using a smaller hashing network (see Fig. 2). Importantly, only the hashing network has to be trained on the data set that the image retrieval is to be used for because the feature extractor network can instead be trained on a different data set which are often already publicly available. In particular, this allows for usage of the large number of available deep learning networks that are pretrained on ImageNet. Since our system outputs binary hash codes the image storage and retrieval is highly scalable. Furthermore the fact that only a small network has to be trained on the data set yields fast training times even for large data sets.
Our MIRS system also allows for both a supervised (Section III-C) and an unsupervised (Section III-D) learning approach. Therefore, it can be optimally used in case that labelled data is available and in case that it is not.
Additionally, our MIRS system uses Inter-class triplet sampling (see Section III-E1) and domain-whitening layers (see Section III-E2) to improve the generalisation capability of the system to different domains.

B. MIRS Hashing Network

As discussed before, for our non-baseline models we use a deep learning network as a feature extractor. On the resulting feature vectors we then train a small network to fine-tune the results and also to reduce the dimensionality of the metric space in order to make storage more efficient. Another advantage of using a small network for fine-tuning is that they also perform well on small data sets. Using a big model to train on small data sets can lead to overfitting which is avoided by this approach. This can be seen in detail in Section IV
The main structure of the hashing network is based on work by Roy et.al. in. [27]
They propose a network composed of three fully connected one-dimensional layers, where the input size of the first layer is equal to the output of the feature extractor network. In between these layers we use a LeakyReLU. The last layer then has K neurons, where K is the desired hash length.
After the last layer, the Sigmoid operator is used on the output in order to restrict the output values to be in the interval $[0, 1]$. The archive is then built by using a simple thresholding operation on the output vector v:

$$b_n = \begin{cases} 1 & \text{if } v_n \geq 0.5 \\ 0 & \text{if } v_n < 0.5 \end{cases} \tag{1}$$

This yields a binary code for every image in the archive set which results in very efficient storage and access capabilities. Image retrieval can then be done using hamming distance as an appropriate distance metric for binary codes.
In summary, the hashing network maps the feature vectors $f \in \mathbb{R}^N$ from the feature extractor to smaller feature vectors $\hat{f} \in \mathbb{R}^K$ which can be transformed into binary hash codes with a low loss of information. [27] Here, N is the number of feature entries in the feature vector that the feature extractor network yields.
As discussed in the previous sections, we use either a triplet loss or a contrastive loss as our main loss function depending on if we are doing supervised or unsupervised learning. However, the exact loss function used for training the hashing

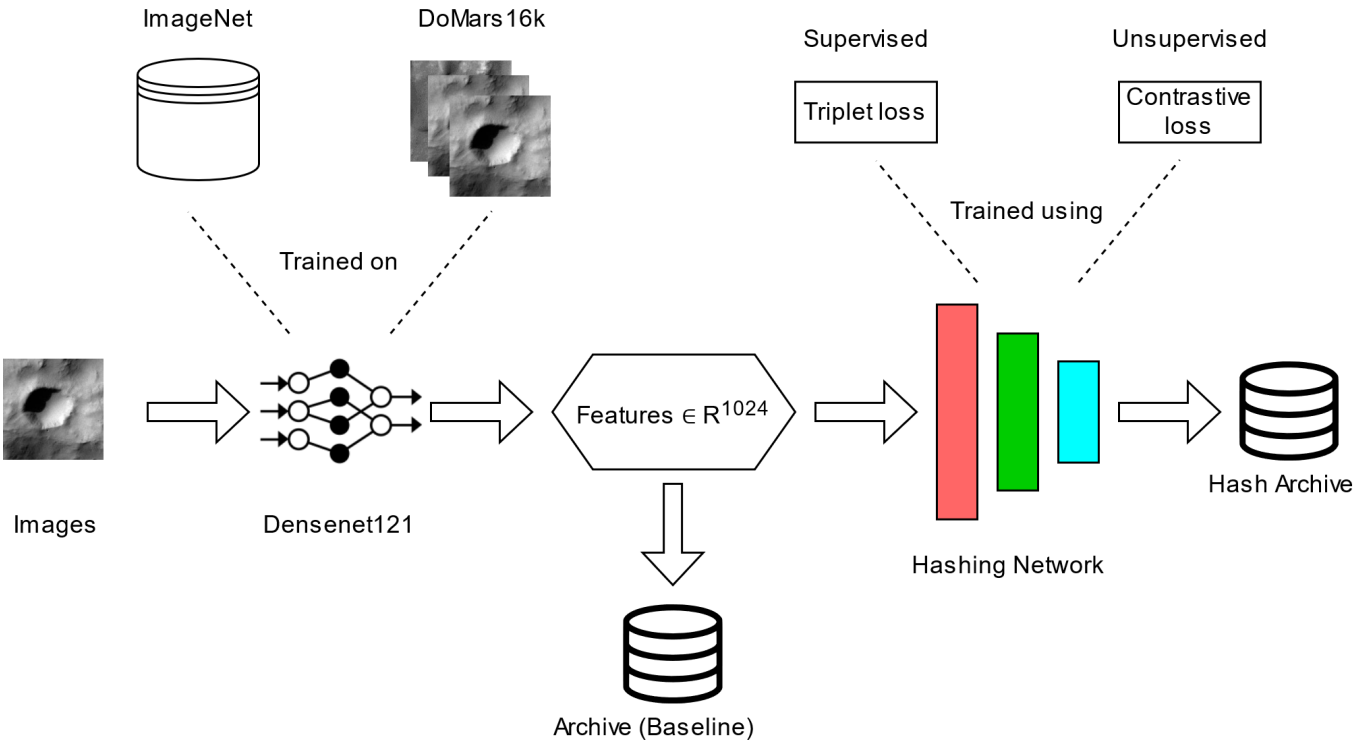

Fig. 2. Complete overview of MIRS architecture and baseline models. The input images are encoded by a DenseNet121 into feature vectors of length 1024. The DenseNet was either trained only on ImageNet or on the DoMars16k data set. The features are fed into a hashing network that compresses them into similarity preserving hash codes. Finally, the hash codes are stored in a hash archive and can be retrieved with a query. The hashing network is either trained supervised with triplet loss or unsupervised with contrastive loss.

network does not only consist of these loss functions.
We also use two additional loss functions that have the purpose of generating well-behaving binary hashes for storage in the archive. [27]
The first of these is the *push loss*. This loss function optimises for values that are far away from the threshold 0.5. Thus the information loss from the thresholding operation is minimised. The push loss is given by

$$L_{push} = -\frac{1}{K}\sum_{i=1}^{M} ||h(g_i) - 0.5\mathbf{1}_K||^2, \tag{2}$$

where h is the hashing network, M is the batch size and $\mathbf{1}_K$ is a one-dimensional vector of length K.
The *balancing loss* influences the hash codes to mostly be either lower or higher than the threshold and therefore results in varied hash codes. The balancing loss function is given by

$$L_{balancing} = \sum_{i=1}^{M}(mean(h(g_i)) - 0.5)^2. \tag{3}$$

The complete loss function is then either given by

$$L = L_{triplet} + \lambda_1 L_{push} + \lambda_2 L_{balancing} \tag{4}$$

for supervised learning or

$$L = L_{contrastive} + \lambda_1 L_{push} + \lambda_2 L_{balancing} \tag{5}$$

in the unsupervised approach. Here λ_1 and λ_2 are hyperparameters that depend on the dataset.

For a deeper explanation of the triplet loss used in MIRS see Section III-C1, for contrastive loss see Section III-D1.
It is important to note that while $L_{balancing}$ and L_{push} are computed directly on the output of the hashing network for the original image, $L_{contrastive}$ is computed on the output of the projection head $g(\cdot)$ (Section III-D1) which is placed on top of the hashing network in the unsupervised approach.

C. Supervised Learning in MIRS

The supervised components of our approach rely on triplet loss to learn the discriminative features between different input samples.

1) Triplet Loss: Triplet loss was first proposed in [28] and is used to discriminate various faces from one another. Images from a labelled data set are sampled in triplets which consist of two images from the same class called *anchor* and *positive* and one image from a different class called *negative*. It is then assumed that the anchor and positive should be semantically closer to each other than the anchor to the negative, since the anchor and the positive are from the same class and should therefore share similar semantic content.
This is characterised by the following equation:

$$\|f(x_a) - f(x_p)\|_2^2 + \alpha < \|f(x_a) - f(x_n)\|_2^2 \tag{6}$$

where x_a, x_p and x_n stand for the *anchor*, *positive* and *negative* respectively and the network encoding is represented by $f(x)$. The margin α serves as a minimal distance by which the output features should be separated.

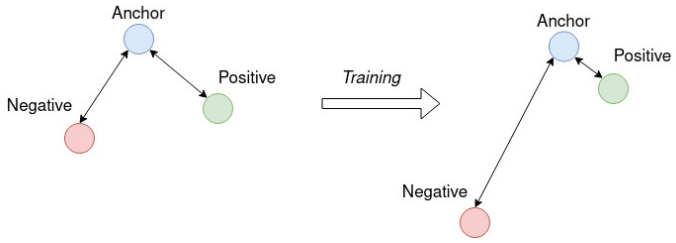

Fig. 3. The working principle behind triplet loss. A feature extractor trained with triplet loss should learn to generate more similar feature outputs for the anchor and positive compared to the feature output for the negative sample.

Based on this formula the overall loss function to be minimised is given by:

$$L_{triplet} = \sum_{i=1}^{N} \left(\|f(x_{ai}) - f(x_{pi})\|_2^2 - \|f(x_{ai}) - f(x_{ni})\|_2^2 + \alpha \right) \tag{7}$$

This simultaneously trains the network to generate a more similar feature representation for images from the same class and less similar features for different classes. This principle is also graphically shown in Figure 3.

However, the performance of the trained feature extractor greatly depends on the selection of triplets from the data set. If the network is mainly presented with easy examples it might not be able to learn what separates more similar classes from one another. This could be solved by only considering hard triplets where for any anchor only the furthest positive feature vector and the closest negative feature vector from all model outputs are selected. This however can oftentimes lead to a collapsed model where the model outputs zeros for all input images because it fails to detect any differentiating features between them.

To circumvent this a relaxation of the sampling constraint is often introduced to mainly sample so called semi-hard triplets with negatives that fulfil the following constraint:

$$\|f(x_a) - f(x_p)\|_2^2 < \|f(x_a) - f(x_n)\|_2^2 < \|f(x_a) - f(x_p)\|_2^2 + \alpha \tag{8}$$

This means that the corresponding feature vectors violate the margin but still contain distinguishable semantic information. Computing these on the whole data set is however infeasible and therefore these are normally selected from within one batch of training samples. Such a sampling strategy is called online triplet sampling and requires sufficiently large enough batch sizes relative to the number of classes. This ensures an adequate amount of feature vectors to select from.

D. Unsupervised Learning in MIRS

Unsupervised learning is based on unlabelled data. Potential tasks could be to find groups, also called clusters, of similar data examples or to determine the data distribution [29]. Naturally, there exist much more unlabelled than annotated data since labelling data in RS is challenging and often requires domain expertise. Therefore, it is advantageous to find approaches that do not need a large amount of annotated training images to achieve good performances.

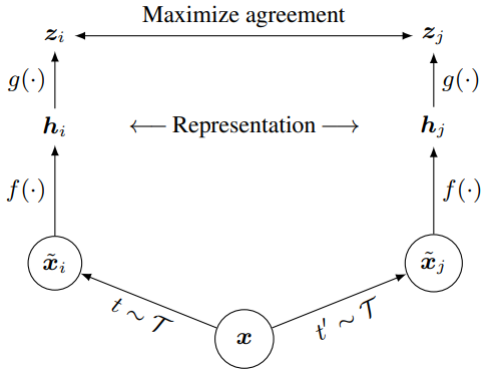

Fig. 4. SimCLR framework. An input example x is augmented by two randomly selected data augmentations t and t'. The correlated examples $\tilde{x}_i$ and $\tilde{x}_j$ are then encoded by a network $f(\cdot)$ and finally mapped into the loss space via the projection head $g(\cdot)$, where agreement is maximised [30].

1) Contrastive Loss: Our unsupervised approach is based on a contrastive learning framework, called SimCLR, introduced by Chen et al., that learns meaningful representations by maximising the agreement between differently augmented views of the same image with a contrastive loss [30].

Figure 4 shows the proposed framework. A given data example x is augmented by two random data augmentations t and t' sampled from a group T of possible augmentations. The resulting correlated examples $\tilde{x}_i$ and $\tilde{x}_j$ are then encoded by a neural network encoder $f(\cdot)$ that computes the feature representations of the correlated examples. Thus, representations h_i and h_j of the augmented views are obtained and finally passed through a projection head $g(\cdot)$, a small neural network, that maps the representations into the space where the loss function is applied resulting in z_i and z_j.

Because of the multi-views a mini-batch of size N now contains $2N$ data points. While a correlated pair is treated as a positive all the other $2(N-1)$ pairs are treated as negatives. The contrastive loss function (also called as NT-Xent [30]) is defined as follows:

$$l_{(i,j)} = -\log \frac{\exp sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp sim(z_i, z_k)/\tau} \tag{9}$$

Where $sim(\cdot,\cdot)$ is the cosine similarity and $\mathbb{1}_{[k \neq i]} \in \{0,1\}$ is an indicator function with $\mathbb{1}_{[k \neq i]} = 1$, if $k \neq i$ and 0 otherwise. τ is the temperature parameter and has to be adjusted. The loss is computed across all (i,j) and (j,i) positive pairs in a batch [30]. We will refer to this loss as $L_{contrastive}$.

We use an approach inspired by SimCLR to train our hashing network (see Section III-B). But, instead of augmenting a given image twice to obtain the multi-views, we augment each image only once with a random transformation. A correlated pair then consists of the original image and the augmented view. Possible transformations are cropping, 90° rotation, horizontal flip and Gaussian noise. We combine a base encoder and the hashing network (Section IV-A4) as an encoder and put a projection head on top. However, we freeze the base encoder and only train the hashing network and projection head, which will be discarded

after training.
Both views (the original and the augmented) are given as input to the model. The contrastive loss is computed on the output of the projection head, while the hashing losses, L_{push} (Equation 2) and $L_{balanace}$ (Equation 3), are only computed on the the output of the hashing network for the first view, which is the orginal sample image. The complete loss is defined in Equation 5.

E. Generalisation Capability Injection

Training a metric-learning system with class-based triplet loss provides a valuable measure for the similarity between images in a data set. However, depending on the specific class properties, there can be images from different classes sharing a high similarity which would not be accounted for by standard triplet sampling strategies. This can leave vital information regarding image similarity unused and could potentially be exploited to increase the performance of our approach.
Furthermore, when training a neural network on a specific data set, the contained data is usually sampled from the same domain which can introduce challenges to adapt a trained model to unseen data sampled from a different domain. As satellite images can vary due to changing light angles, weather or ground conditions, it can improve performance greatly to apply domain adaption techniques to ensure high performances on unseen target data. To increase the generalisation capabilities of our MIRS model and circumvent the implications of the aforementioned problems we introduce two separate modifications to the architecture and training process .

1) Inter-class Triplet Sampling: Ranking losses based on class label information, like the triplet loss as described in III-C1, can only discriminate between images in terms of similarity based on the given class information. However, class labels do not convey all information required to identify and distinguish between relevant content in images which means that the quality of the resulting ranking partly depends on the selection of the class labels. This is especially relevant because most available labelled data sets are intended to train a classifier for a specific classification task and might therefore not optimally represent all information required to train a CBIR system.
To mitigate such effects on our MIRS system we utilise a method to introduce further information not included in the original class labels into the training process.

This approach has been proposed by Milbich et al. [31] who have developed a method to identify new features which are shared by multiple images from different labelled classes and therefore form separate classes altogether. These new groupings are intended to complement the labelled training data by presenting the hashing network with new information for the same images. Due to the problem definition these new groupings must be generated in an unsupervised manner and the authors have proposed the use of clustering for this task. However, this introduces the problem that most clustering algorithms would also converge to a grouping similar to the class label assignment. To circumvent this effect before computing the clusters, all feature representations are standardised according to the following formula:

$$f_c^i = \frac{f_c^i - \mu_c}{\sigma_c} \tag{10}$$

with f_c^i representing the feature vector corresponding to the i-th input image belonging to class c. μ_c and σ_c are the mean vector and the diagonal of the co-variance matrix respectively and are computed over all feature vectors from class c.
The clusters are then generated based on these standardised feature vectors over the entire data set. The resulting clusters however still contain many feature vectors belonging to the same labelled classes, because feature standardisation can not ensure that all class specific information has been eliminated. To prevent these overlapping collections of feature vectors from affecting the training process an additional constraint is introduced for the sampling of triplets from the clusters. This means that when triplets are sampled based on which cluster they belong to, it will also only consider such triplets where each constituent belongs to a different class in the original labelled data set. This ensures that all discriminative information represented by those triplets is not already contained in the original classes and in fact presents new information extracted by the clustering process.
These so called inter-class triplets can then be utilised in the training process by calculating the same triplet loss as described in III-C1 for their feature outputs. The resulting loss is then simply added to the loss computed over the class-based triplets.

2) Domain Whitening Transforms: While Inter-class triplets can convey vital information already present in the training data set, also called the source domain, it does not increase the generalisation capabilities of the system when presented with data from an entirely different target domain. This could for example be images taken from different angles or on other spectral bands. Therefore to make it possible to adapt our current approach to such new data we integrated a domain adaption technique called Domain Specific Whitening which was conceived by Roy et al. in [10]. This method is based on the concept of batch normalisation in that it applies a similar transformation on each feature vector. However, instead of the standardisation of the features in batch normalisation this is replaced by a new whitening transformation.
This *batch whitening* is defined in the following formula:

$$BW(x_i, \Omega_B) = \gamma * \hat{x_i} + \beta \tag{11}$$

$$\hat{x_i} = W_B(x_i - \mu_B) \tag{12}$$

With x_i being the i-th vector of a batch and $\Omega_B = (\mu_B, \Sigma_B)$ referring to the mean vector and co-variance matrix computed for a specific batch B, W_B is the result of a decomposition of Σ_B according to the following formula:

$$\Sigma_B = W_B^T W_B \tag{13}$$

The precise process for calculating W_B is described in detail in [32] but would exceed the scope of this work. The γ and β are a simple scale and shift parameter respectively.
The Domain Whitening Transform (DWT) is then defined to apply the batch whitening operation BW on all input features. This transform is integrated into our approach by applying it to the encoders feature output that is supplied to the hashing network.
To estimate the Ω of the DWT for the inference it calculates an estimate of the mean vector and the co-variance matrix during training by averaging over multiple batches of the respective data set. This is required because the quality of the outputs from the DWT relies on an accurate estimate of the specific Ω for a domain. This is achieved for both the source and the target domains by utilising two separate DWT layers for each and inserting one during the normal training of the hashing network and training a separate one on the target domain using a different loss function. The DWT trained on the source domain is then discarded after training and replaced in the inference phase by the DWT trained on the target domain containing the accurate Ω estimates for its respective images.
To ensure the effectiveness of the DWT layer the authors of [10] also provided a simple entropy loss function to be applied on the transformed features. This entropy loss is defined as follows:

$$L_{entropy} = -\frac{1}{m}\sum_{i=1}^{m}\Big(softmax(x_i) * \log(softmax(x_i))\Big) \tag{14}$$

$$softmax(x) = \frac{\exp(x)}{\sum_{j=1}^{n}\exp(x_j)} \tag{15}$$

Where m is the batch size, n is the length of the feature vectors and $*$ defines the element-wise multiplication of two vectors. By integrating this DWT layer and the specified loss function into our approach we hope to achieve a better generalisation of our model to different domains of mars images.

IV. Experimental Results

A. Experimental Setup

1) Evaluation Procedure: The approach to evaluate the performance of the different components described in this paper is based on the structure proposed by Wan et.al in [33]. They introduced three different schemes representing different compositions of the components of their model and evaluated the added components by comparing them to the simpler schemes. Corresponding to their SCHEME-1 type model we define a baseline approach where we directly use the features from a Densenet121's last layer that was either pretrained on ImageNet, trained for classification on [34] or trained with triplet loss to optimise the distance between the output features. These simple approaches will be used to evaluate the capabilities of differentiating between the provided images solely based on the feature output.
Integrating the hashing network with these feature extractors is then similar to the SCHEME-2 because we train a small additional network to refine the result of the pretrained Densenets and also bring it into a form in which the features can be stored in an efficient and scalable way. The following sections will first introduce the data set on which we trained our models and then give a description of the specific configuration of static parameters used for each of the components. This will be followed up with a sensitivity analysis of variable hyper-parameters used in the training process also separately analysed for each component.
Finally all components will be combined and compared to each other as well as the baseline approaches to evaluate the general performance of our approach as well as the improvements introduced by each component.

2) Data set: The data set we are using is the DoMars16k data set. [34] It consists of 16150 patches of the surface of Mars that were manually created by experts from 163 big CTX images. Each patch is a 200 px $\times$ 200 px single-channel image which corresponds to 1.2 km $\times$ 1.2 km on the Mars surface. Every patch contains one of fifteen landforms which results in approximately 1076 images per class. We split the data set into four parts, using 35% for training, 35% for the archive set that we run the queries against, 20% for the validation set and finally 10% for the test set.
Important to note is that the classification of the patches was not always done in an exhaustive way. This means that there can be very visually and geographically different images within the same class. For example, dunes were categorised into two groups, "Aeolian Straight" dunes and "Aeolian Curved" dunes. "Aeolian Straight" dunes refer to dunes that were formed by a single wind direction and are therefore linear in appearance. "Aeolian Curved" then refers to dunes that were formed by more complex winds and therefore appear non-linear [34]. However, this dune classification is not as specific as the standard classification scheme for dunes given in [35]. Because of this, there are multiple different types of dunes that were all classified as "Aeolian Curved". Three of these examples can be seen in Fig. 5. There also exists the

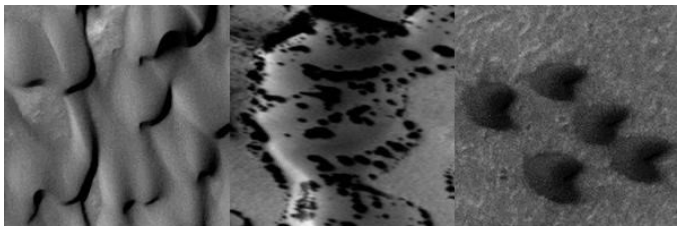

Fig. 5. Different images of dunes from the DoMars16k data set.

exact opposite case of this where two images are visually very similar but were labelled with two different classes. This case appears mostly within the greater category of "Basic Terrain Landforms", which consists of the classes "Mixed Terrain", "Rough Terrain", "Smooth Terrain" and "Textured Terrain". "Mixed Terrain" consists both of patches with rough and smooth terrain. [34] This leads to the fact that there are

”Mixed Terrain” images that are very similar to either ”Smooth Terrain” images or ”Rough Terrain” images. An example of this can be seen in Fig. 6. Another obstacle for training a

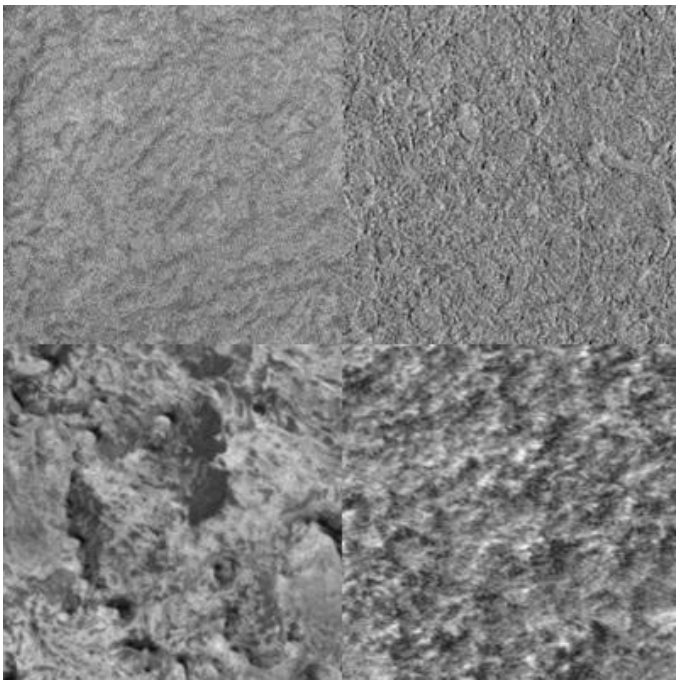

Fig. 6. Example images from the DoMars16k data set. Left: Mixed Terrain, Right Top: Smooth Terrain, Right Bottom: Rough Terrain; Enlarged to better visualise differences and similarities

content-based image retrieval system using this data set is the occurrence of images that partly contain land forms from another class. There are for example images in the classes of ”Ridge” and ”Cliff” that contain small craters. A few examples are shown in Fig. 7. These three types of classifications are

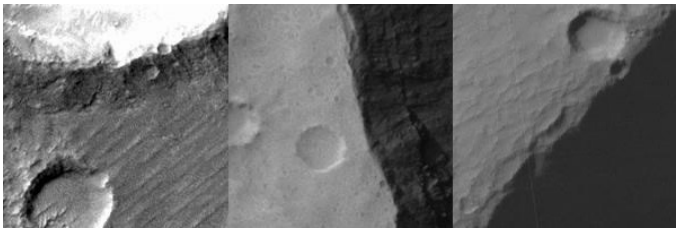

Fig. 7. Ridges and cliffs containing craters from the DoMars16k data set.

problematic for training and also for verifying the performance of a content-based image retrieval system, which we will discuss in detail in Section IV.

3) Basic Pipeline: In order to obtain a baseline to compare the results of our more complex Content-Based Image Retrieval (CBIR) systems to, we first implemented a basic CBIR pipeline. For this we used a Densenet121 which was pretrained on our training subset of the DoMars16k data set. We then removed the classification layer in order for the network to return the feature vectors. Keeping the classification layer would result in simply finding images from the same class regardless of visual similarity. The feature layer however might have information about more than just class correspondence.
We then applied this modified Densenet on every image in the archive set and saved the resulting feature vectors along with paths to the images they belong to as well as their class label in a database. For the database we used a simple json file since the data set was small enough that performance and size were not of concern for our basic pipeline.
We can now query the most similar images for a given image by applying the modified Densenet on this image and comparing the resulting feature vector to every feature vector in the database by their euclidean distance.
This of course is a very simple architecture. However, the results were already surprisingly good as can be seen in Section IV.
The biggest issues of this approach are as follows: First, since the feature vectors are composed of a large number of floating point numbers, this results in a very big archive database. For our very small data set the archive was already 120 Mb big. This also means that this approach is not usable for bigger data sets.
Second, since the network was trained with the purpose of classification and not image retrieval, it will mostly just find images of the same class and will not be good at finding images by visual similarity.
Third, this network does not have a capability to generalise to other data sets. If the other data set is labelled, then it would be required to completely train a Densenet on this data set. If the other data set is unlabelled, then it is impossible to use this approach at all since we cannot train a classification network on unlabelled data with this setup.
All of these points are addressed by our MIRS architecture.

4) MIRS implementation: The architecture based on MiLaN [27] is the main model structure which is described in Section III. For the deep learning network that is used as a feature extractor we use a DenseNet121, as in the baseline. The hyperparameters used for the hashing network are $\lambda_1 = 0.001$ and $\lambda_2 = 1$ and a batch size of 256. We also use the Adam optimizer with $\beta = (0.9, 0.99)$ and a margin of 0.2 for the LeakyReLU layers.
To improve the training speed, we used the Densenet on every image in the archive set once at the beginning of training and saved the resulting feature vectors in memory. For the training we can then simply access these feature vectors instead of having to use the Densenet again in every epoch.

5) Supervised Approach: For the supervised approach the MIRS architecture is trained using standard online triplet loss as it is described in Section III-C1. For this we relied on the python module pytorch-metric-learning which contains an implementation of fast online triplet sampling when provided with the class labels and feature outputs. For the sampling strategy we selected semi-hard sampling because easy sampling would not present enough discriminative information and hard sampling resulted in a collapse of the model so that all outputs where zero. The margin for the sampling was set to be 0.2 as this is the commonly used margin value throughout the

literature. To compute the distance between the feature vectors we selected the euclidean distance as it is also commonly used throughout the literature.

6) Unsupervised approach: As an encoder we use the same architecture and hyperparameters as described in Section IV-A4 and place a projection head on top of the feature output from the model (see Section III-D). The projection head consists of two fully connected layers and a ReLU. The output dimension is adjustable and needs to be determined with experimental results. After training the projection head is discarded.
The temperature parameter, which is used to compute the contrastive loss (Equation 9), τ is set to 0.1.

7) Inter-class triplets: The k-means algorithm is applied on all outputs of the feature extractor which are stored in a local data structure. The output vectors created for each input image by the feature extractor are grouped into different clusters by the k-means algorithm. We use sklearn's implementation of this algorithm with a number of clusters k that needs to be determined experimentally. The inter-class triplets are then directly sampled in the same manner as the standard triplets with the same parameters as described in Section IV-A5. However, instead of the class affiliation they are sampled based on group affiliation in the results from the clustering step. Afterwards all triplets are removed which violate the constraint laid out in Section III-C1 that all triplet constituents must be from different classes in the original data set. The resulting triplets are used to compute an additional triplet loss which is then added to the overall loss.

8) Domain Whitening Transforms: The implementation of the Domain Whitening Transform and the Entropy loss are heavily based on the original implementation provided by [10]. However their 2D transform has been adapted to a 1D transform to conform with the feature outputs from the feature extractor network. It is loaded as a separate layer and directly applied on the feature outputs. This allows it to be activated separately from the main hashing network. The main hyper-parameter influencing the DWTs performance is the group size used for the estimation of the batch statistics and needs to be determined experimentally. The resulting whitened feature vectors are then used to calculate the Entropy loss as described in Section III-E2.

9) Mean Average Precision: We used mean Average Precision (mAP) as a metric to evaluate the performance of our baseline and MIRS systems because it is the established standard metric to measure the performance of content-based image retrieval systems [6] [9] [11]. It is defined as the mean over all average precisions (APs). In our case of image retrieval this means that for every image in the test set we do a query q using our system. Then the mAP is defined as

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}. \tag{16}$$

Here Q is the number of queries and therefore identical to the number of images in the test set. Assume that a query q has a total number of N results and among them P positives. A positive in this context is defined as an image of the same class as the queried image. The average precision is then given by

$$AP(q) = \frac{1}{P} \sum_{k=1}^{N} \mathrm{P@k} * \frac{1}{k}. \tag{17}$$

In our experiments we always set $N = 64$.
P@k is a function that returns 1 if the image k is a positive and 0 otherwise. Average Precision therefore rewards correct results more if they appear earlier in the query result. This makes it a very good metric for measuring how well an image retrieval system finds images of the same class. The reason for using this metric for content-based image retrieval is the idea that images of the same class are more likely to be visually similar then two images of different classes.
It is however important to note that this metric is not perfect, since especially for our DoMars16k data set, there can be both very visually distinct images within the same class and visually similar images from different classes. Therefore for our result analysis between different approaches we also rely on manual visual comparisons of the result queries instead of only using the mAP. Doing this automatically would not be possible since our data set does not contain ground truth similarity data.

B. Sensitivity Analysis

The performance of our proposed hashing approach regarding similarity of the retrieved images to a query image depend heavily on the selection of specific hyperparameters. To evaluate the effects of the variation of these parameters on the model performance we conduct a thorough sensitivity analysis.

1) Hash Code Length: The length of the hash codes outputted by the hashing network can effect the general performance of the image retrieval system. This is because the amount of bits available determines the amount of similarity information that can be encoded per image. Therefore, it is to be expected that the performance should rise with a higher hash code length as the network can represent more variation within the different bit combinations. However, this would also increase the resulting size of the hash code archive which means that a trade-off between hash code length and performance is necessary to select the optimal length parameter. As expected, the performance improved

TABLE I
DIFFERENT HASH CODE LENGTHS FOR THE SUPERVISED APPROACH WITHOUT GENERALISATION INJECTION; CHOSEN LENGTH IS HIGHLIGHTED.

Hash Code Length	mAP@64
32 Bits	0.746
64 Bits	0.757
128 Bits	**0.767**
256 Bits	0.767

with rising hash code length. However, from 128 Bits to 256 Bits there was no further improvement, indicating that 128 Bits are sufficient to represent the learned information about the images. Since our data set was quite small, even for a hash code length of 128 Bits the archive was very small. Therefore, we use 128 bits as the hash code length going forward for slightly higher precision. If the used data set was larger, it might have made sense to use a smaller hash code length to reduce the archive size with low loss of performance.

2) Projection dimension for unsupervised approach: Table II shows the results for different projection head output dimensions. The projection head is placed on top of the hashing network, which outputs a 128-bit hash code. A projection dimension of 128 yields the best mAP with 0.619. In this case the number of features in the layers of the projection head does not change, which means that simply adding a non-linearity after the hashing network leads to the best results. Dropping the projection dimension to 64 and 32 lead to a mAP decrease of 0.022 and 0.027 respectively. This drop is to be expected since information are lost when applying contrastive loss in a lower dimension. However, increasing the dimensions any further than 128 does not improve mAP either, which shows that there is no value in adding any more complexity to the model and that a simple non-linearity is sufficient.

TABLE II
HASHNET RESULTS WITH VARYING PROJECTION DIMENSION FOR THE UNSUPERVISED APPROACH

Projection dimension	mAP@64
32	0.592
64	0.597
128	**0.619**
256	0.605
512	0.614

3) Number of clusters k for Inter-class triplet sampling: The number of clusters k determines the amount of groups to which the output features of the DenseNet encoder are assigned before the training process. This influences the sampling of inter-class triplets as the amount of clusters specifies how many different feature variations can be represented by a separate group. The number should be higher than the number of labelled classes to ensure that the clustering does not just repeat the class distribution but should also not be too high to prevent the clusters from becoming to small to represent meaningful information to the network. Table III shows the results obtained using the standard hashnet architecture and the ImageNet DenseNet as the encoder network. The hash length remained fixed to 128 bits. The mAP score over all cluster sizes is significantly lower than without inter-class triplets and they all lie in a similar range. The best score however was achieved with a cluster count of 30 which we will be using in further tests.

4) Group Size for Domain Whitening Transform: The group size for the DWT layer controls the sub-selection of features for a given encoding output for which the co-variance matrix will be computed. This is important because using the full range of features can result in a ill-conditioned co-variance matrix and as a consequence in a lower performance. This effect can also be seen in Table IV as the performance with integrated DWT layer improves significantly for smaller group sizes but then becomes relatively stable at group sizes between 16 and 128. Based on these results we select a group size of 32 for the further comparisons to the other components as it resulted in the highest increase in performance.

TABLE III
HASHNET RESULTS WITH INTER-CLASS TRIPLETS

N clusters	mAP@64
15	0.635
20	0.673
25	0.663
30	**0.678**
35	0.595
40	0.592

TABLE IV
HASHNET RESULTS WITH VARYING GROUP SIZE FOR DWT

Group Size	mAP@64
1024	0.642
512	0.703
128	0.779
64	0.784
32	**0.791**
16	0.786

5) Different Densenets: Supervised Approach: In this section we compare three differently trained DenseNet121s as the feature extractor networks for the same hashing network, specifically the supervised hashing network without generalisation injection. The first feature extractor network was simply trained on ImageNet without further fine-tuning, the second was trained on the training subset of DoMars16k for classification purposes as in [34] and the third was trained on the same data set using triplet loss.

TABLE V
VARYING DENSENETS FOR HASHING NETWORK

DenseNet121 trained on	mAP@64
ImageNet	0.767
DoMars16k for classification	0.819
DoMars16k using Triplet Loss	**0.884**

As can be seen in Table V the mAP score is higher for the DenseNets trained on the DoMars16k data set. This was expected, since the classification DenseNet was specifically trained to separate the data samples into specific classes, which serve as the ground truth for the mAP computation. Consequently, this results in a higher mAP score. The same principle explains why the mAP is also higher for the DenseNet trained on DoMars16k using triplet loss since triplet loss also uses the same classes for the choice of the triplets as are used for the mAP. Since mAP could be skewed for these reasons, we

also compared the results visually.

As the first query image in Figure 8 shows, the system with the DenseNet trained on ImageNet prioritises the classes from the DoMars16k data set slightly less, which is why the result consist of craters even though the query image is an aeolian curved dune. However, the craters do look visually very similar to the query image. This is likely because for this system the DenseNet has no information about the DoMars16k data set and the ImageNet is assumed to be uncorrelated to the Mars images. Consequently, the generated feature space separates images not based on the DoMars16k classes but rather only on general visual similarity. Since the hashing network was trained on DoMars16k the system still learned some information about the classes, it is however expected to be less than systems where both the feature extractor and the hashing network are trained on the DoMars16k data set.

Indeed the system using a DenseNet trained on DoMars16k with triplet loss almost never finds images from different classes which can also be seen in the high mAP score. This leads to results that are not visually similar as for example for the first query image, since for this query image there exist no similar looking aolian curved dunes images in the whole data set. The DenseNet trained on DoMars16k for the classification task shows results that are in between the two other models. It finds an aeolian curved dune as the first result but also finds visually similar craters as well. This is within expectation because training for classification does not separate the different classes as strongly as training using triplet loss does, since generating a metric space with large distances between different classes is the whole purpose of triplet loss.

However, the approach using a DenseNet pretrained on DoMars16k also has advantages. While it is not good at finding visually similar images from different classes, it is very good at finding visually similar images within the same class if they exist. This can for example be seen in the second query image, where the first result the models pretrained on DoMars16k finds is an almost identical image that is not found by the system using a DenseNet trained on ImageNet. This can also be seen for the third query image where it is slightly better at finding other small craters whereas the other approach finds both small and large craters alike.

As discussed earlier, while the DenseNet trained on DoMars16k for classification prioritises finding images of the same class more than the DenseNet trained on ImageNet, it does so less than the DenseNet trained on DoMars16k using triplet loss. This is also illustrated by the fourth query image, a ridge containing a small crater, where only the latter finds other ridges with small craters within, while the other two models only find craters which do have visual similarity since the image contains a crater but are not within the same class. It is important to note that these differences are only this pronounced for images that are hard classify because there exist either no similar images within the same class or very similar images in different classes. For most other images all three approaches yield very good results.

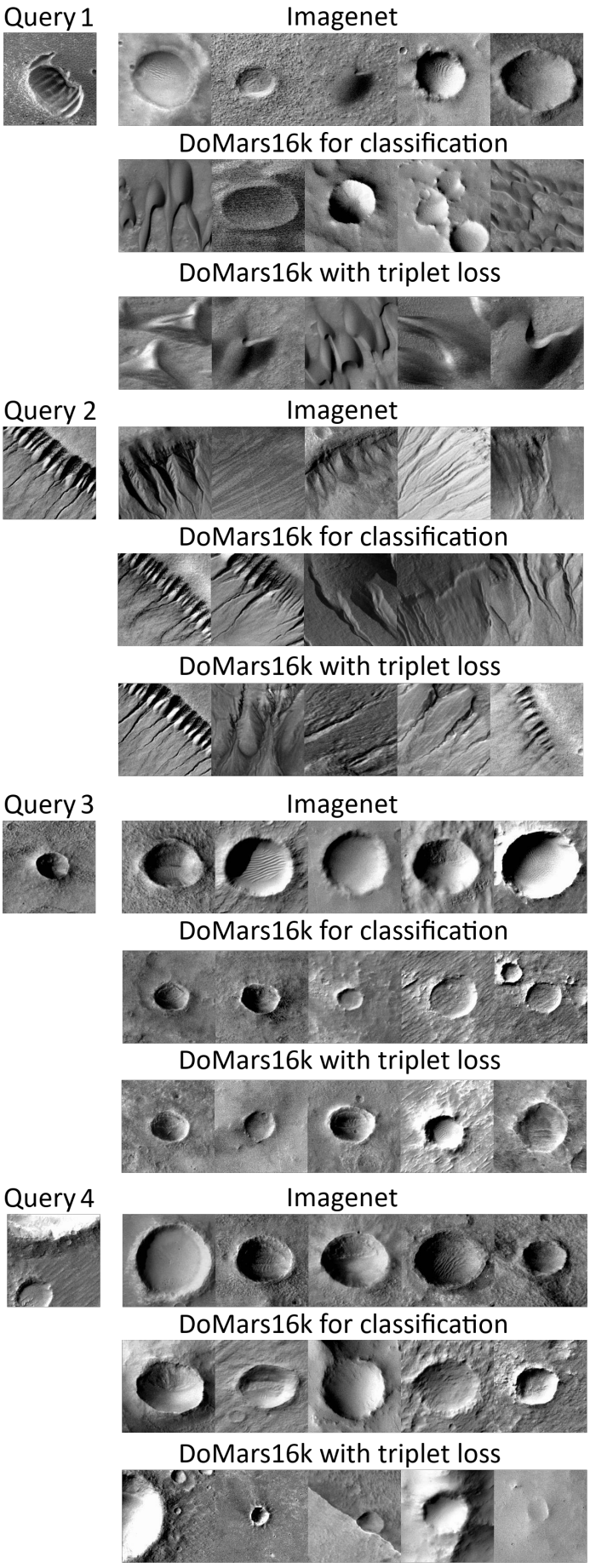

Fig. 8. Visual results for four example images showing the effect of varying the DenseNet type as feature extractor for the hashing network.

To also verify that the performance differences observed when varying the DenseNet type are replicated in the hashnets with inter-class triplets and DWT layers, we conducted further experiments with both of these approaches. The results can be found in Table VI and show that the effect on performance can still be noticed as all configurations still show some form of performance decrease from triplet trained DenseNet to the bare ImageNet trained DenseNet. This further implies that the features provided by these three feature extractors allow a better discrimination in terms of class affiliation which is also noticeable when applying our different configurations for training the hashing network.

TABLE VI
VARYING DENSENETS FOR HASHING WITH DWT AND INTER-CLASS TRIPLETS

DenseNet121 trained on	mAP@64 with inter-class triplets	mAP@64 with DWT
ImageNet	0.677	0.791
DoMars16k for classification	0.819	0.860
DoMars16k using Triplet Loss	0.877	**0.889**

C. MIRS Model Comparison

Table VII shows the results of different MIRS configurations with DenseNet trained on ImageNet: Supervised MIRS and Unsupervised MIRS without Interclass and DWT, supervised MIRS with DWT, but without Interclass and vice versa, as well as unspervised MIRS with DWT.

TABLE VII
DIFFERENT MIRS MODEL CONFIGURATIONS WITH DENSENET TRAINED ON IMAGENET

MIRS Model Configuration	mAP@64
Supervised	0.767
Supervised with DWT 32	0.791
Supervised with Interclass	0.678
Unsupervised	0.619
Unsupervised with DWT 32	0.617

1) Unsupervised MIRS: It is noticeable how the unsupervised configurations score a significantly lower mAP than the supervised configurations. This is partly to be expected since mAP is a measurement for how well a retrieval system finds images of the same class. A supervised system is trained with annotated data and learns with the help of the data labels, while an unsupervised system learns without class labels. Naturally, the mAP will be higher for a supervised approach and thus the mAP is not a very expressive metric in general for the unsupervised approaches. However, when comparing the unsupervised approach with and without the DWT layer it can be seen that it does not change the mAP score to a noticeable degree, while it did increase the mAP for supervised MIRS by 0.024. Therefore to evaluate the effect of both training procedures and the effect of the DWT layer it is necessary to compare visual results.

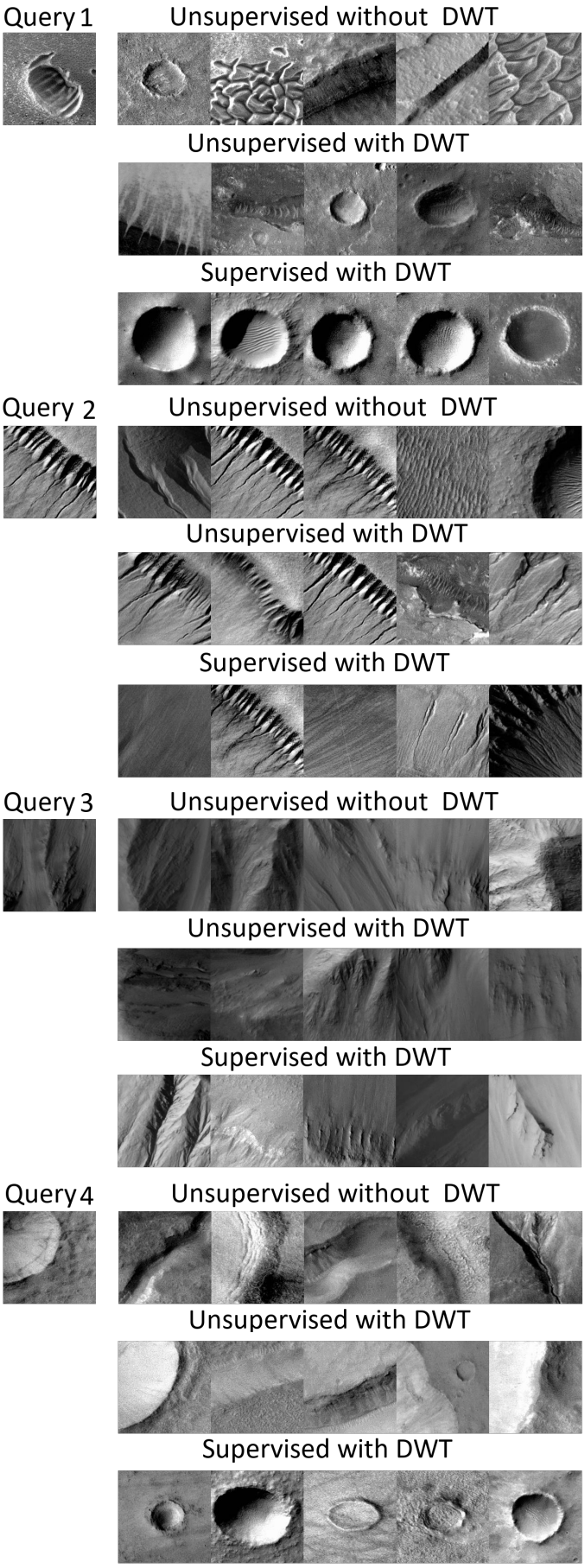

Fig. 9. Comparison of unsupervised approach without domain whitening transforms, unsupervised approach with domain whitening transforms and the supervised approach with domain whitening transforms

Figure 9 shows example queries for the unsupervised MIRS, the unsupervised MIRS with a DWT layer and the supervised MIRS with a DWT layer.
When comparing these example results it can be noticed that even though the mAP does not change significantly when adding the DWT layer to the unsupervised approach it still leads to very different ranking results. However, while these sometimes contain visually more similar results to the query, as can be seen in the examples for query 2 and and 4, for other images the exact opposite seems to be the case as can be observed for query 1. This hints at a certain level of effectiveness of the DWT layer in this configuration for specific types of images, but since this seems to conflict with the results for other images this can not be conclusively interpreted as a positive effect.
Additionally, when comparing the images both unsupervised approaches output a significant amount of visually very dissimilar images to the query. For example, for query 2 with outputs 4 and 5 without a DWT layer and outputs 4 with a DWT layer. Compared to this for the same query both approaches also find very similar images in the other outputs. This might be because the approach is sensitive to similarity of detailed structures within the images. Compared to this the outputs for the supervised approach are more consistent in sticking with one class for the output ranking. But mistakes in the detection of the class affiliation for the query can therefore severely impact the output rankings quality. This is for example visible in the results for query 1 and 4 where the supervised approach classifies both as craters but they actually belong to different classes in the original data set. Its interesting to note here that for both of these the unsupervised approaches return results that belong to the query image's class. However, the results for query 1 and 4 in Figure 8 show that this effect can be circumvented by utilising a DenseNet trained with triplet loss which better encodes such differences in its output features.
It should however be noted that the results for the supervised approach in this image do not represent the usual quality achieved by the supervised implementations as will be apparent from the further analysis provided in this section.
To conclude the analysis of the unsupervised approaches we discuss two potential reasons why the mAP might however still not change significantly when training with a DWT layer: First, domain whitening has the effect that the model becomes more robust to inherent differences between the source and the target domain and therefore better prevents overfitting on the source domain. Thus, it could have a greater effect on the supervised MIRS, which is more prone to overfitting than the unsupervised MIRS which utilises heavy data augmentation. Additionally, the data augmentation virtually increases the training data set size which further decreases the risk of overfitting.

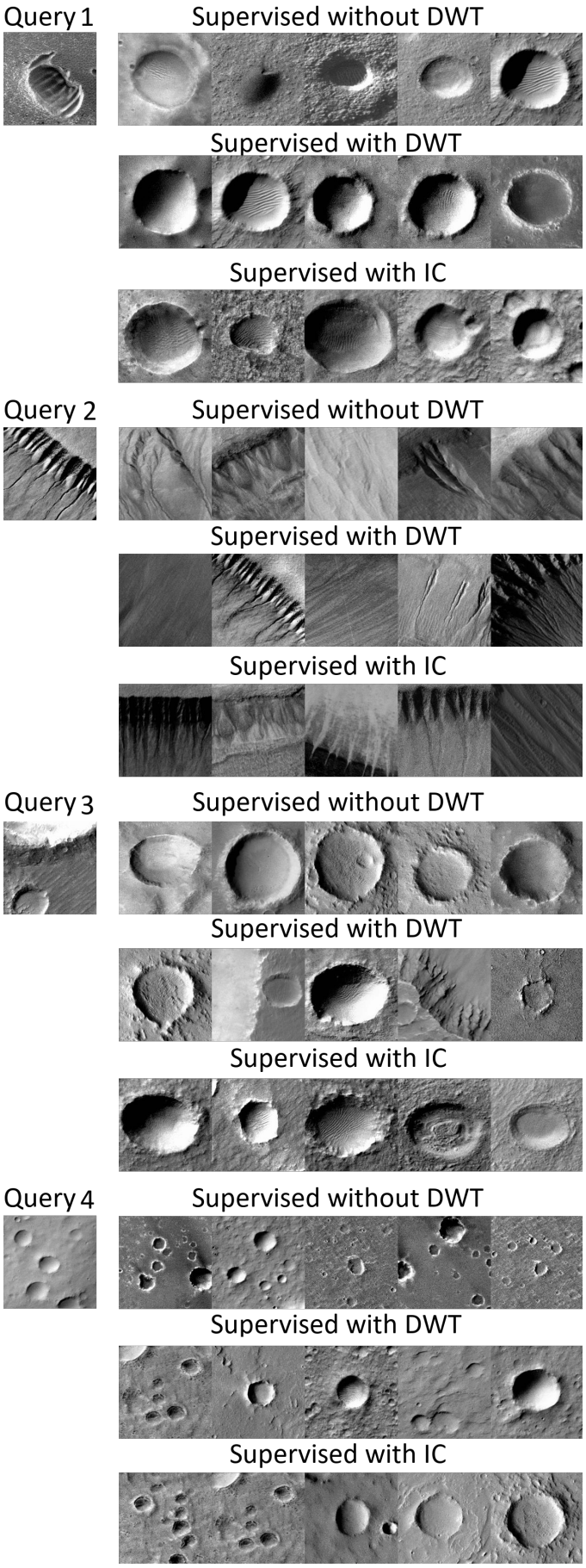

Fig. 10. Comparison of different supervised approaches: The first uses neither domain whitening transforms nor inter-class triplets, the second uses the former and the third uses the latter.

Another possible reason could be that unsupervised MIRS is not aimed to determine the discriminative structures within the data set like triplet loss does, but aims to maximise agreement between similar images. The DWT layer however aims to reduce the impact of certain features from the feature extractor output on the the target domain by utilising the underlying statistical structure of this data set. This might not effect the unsupervised learned model as much because it was trained to achieve a direct similarity between two samples and was not provided with any high level structural information about the whole data set as it was the case for the supervised approach.

2) Supervised MIRS: When comparing the mAP scores for the supervised MIRS configurations in Table VII it is obvious that they generally achieve higher mAP scores than all other configurations. However, while introducing the DWT layer to the model increases the mAP score it decreases with the introduction of inter-class triplets. Though, this is to be expected since introducing new triplets which contain similarity information not present in the original labels would naturally decrease the mAP score as it is calculated on these original class labels. Therefore, we analysed some visual examples to further determine the effect of both generalisation methods. Figure 10 shows various example results for different supervised MIRS configurations with and without added generalisation capability. For all of these examples the DenseNet trained on ImageNet served as the feature extractor. Here it can be seen that all three approaches produce results which seem relatively similar to one another and especially the results for training with inter-class triplets contain no obvious wrong results in contrast to what would be expected due to the low mAP score. However, the degree of similarity between the query and the retrieved images varies noticeably for all approaches. In most of these examples the different models categorise the query into the same class even though this categorisation is sometimes not correct as can be seen for query 1.
But the images returned by the model with a DWT layer often fit the query slightly better than the other approaches. For example, in query 1 where its outputs better estimate the brightness gradient from the query image. Another good example is query 3 where it returns images with both a ridge and a crater, while both of the other approaches can only determine the crater and thus only output crater images in the ranking.

In contrast, for query 4 only the unmodified configuration returns the ground truth class of the query image, which is a crater field, but both generalisation modifications return more craters in their outputs. This could however be explained by the fact that the various craters visible in the query have more similarities with the singular craters than the other crater fields.

This could in general be a possible explanation for the lower mAP score of the model with inter-class triplets as such a relationships in the data might receive stronger emphasis by the clustering algorithm and therefore be introduced in the training.

While it seems that the DWT layer also leads to more results from different classes for the shown examples this conflicts with the higher mAP score calculated over all output rankings. This means that for most other samples in the test set the output rankings must belong to the same class as the query meaning that this effect is likely more pronounced in these examples then it is in the whole data set.

However, when employing different DenseNets for the feature extraction step, as has been explored in Section IV-B5, classification accuracy can be greatly improved. This becomes directly apparent when comparing query 1, 2 and 3 with the results in Figure 8. The base supervised approach with a feature extractor trained on the data set returns visually much more similar results than any supervised approach based on the ImageNet feature extractor. Taken together, these findings suggest that the hashing network can only perform a similarity differentiation to a certain degree and greatly relies on the feature extractor for more complicated connections within the data set as will be further explored in the next section.

D. Baseline Comparison

We will now compare the performance of our hashing approach and the various modifications we integrated to the baseline CBIR system described in IV-A3.
As shown in Table VIII the unsupervised approach with DWT has the lowest mAP of all models. However, as discussed in detail in Section IV-C1 the model nonetheless has advantages over the other systems in some areas.

TABLE VIII
RESULTS OF THE BEST UNSUPERVISED AND SUPERVISED HASHNET MODELS AND THE BASELINE

	Baseline	Supervised without DWT	Supervised with DWT	Unsupervised with DWT
ImageNet	0.668	0.767	0.791	0.617
DoMars16k for classification	0.844	0.819	0.860	-
DoMars16k using triplet loss	0.871	0.884	0.889	-

The supervised approach improves the mAP of the baseline for all three tested types of feature extractor networks and also both with and without the usage of domain whitening transforms. This indicates that the inclusion of the hashing network not only maps the features obtained with the feature extractor network well into the lower-dimensional hash space which would result in an equal or slightly lower performance. Instead, it seems to even improve the quality of the features slightly during the learning process.
As we discussed in previous sections, mAP alone is not a perfect metric for evaluating performance of our systems because of the particularities of the classification in the DoMars16k data set. Consequently, we also compared the results visually. To illustrate the results of this process, we selected two images that show the similarities and differences between the different systems well. These can be seen in Figure 11. All three systems compared in this figure use the DenseNet121 trained on DoMars16k with triplet loss as the feature extractor. The

results for the other feature extractors were however analogous but overall slightly worse which is what the mAP scores would suggest.
The only exception to this is the baseline trained on ImageNet for which the result was closer to the results observed in the unsupervised MIRS system. This was expected since this baseline was not trained on the DoMars16k data set at all and therefore also constitutes an unsupervised learning scenario. This is also reflected in the mAP score which is closer to the mAP score observed for the unsupervised system.
For the systems using the DenseNet121 trained on DoMars16k with triplet loss we noticed that the baseline performed visually very similar to the MIRS system without DWT as can be seen in both queries in Figure 11. In the second query image the MIRS system without DWT even produces a marginally better result since the first two images in the result are extremely similar which is only the case for the first retrieved image in the baseline system. We did not find enough evidence however to definitively conclude that the MIRS without DWT performs better than the baseline overall. What is clear is that the MIRS without DWT is at least as good as the baseline system while also having the benefits of scalability and much greater training and retrieval speed as we will discuss in detail in Section IV-E.
The MIRS system with DWT visually performed worse than both the baseline and MIRS without DWT in some cases as is seen in both examples in Figure 11. For most images the results were very similar however for a few images the MIRS system using DWT even produced better results. The mAP suggest that it also was the best system at retrieving images from the same class as the query image. Overall the results concerning the DWT matched the observations made in Section IV-C, so we will not go into further detail on this topic here.

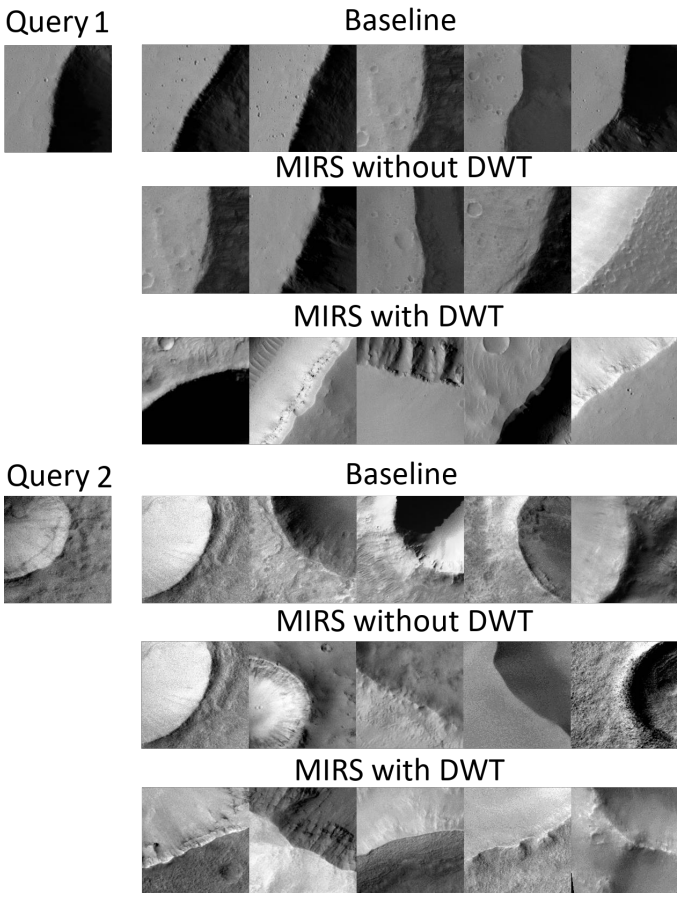

Fig. 11. Comparison of three models trained using the DenseNet121 trained on DoMars16k using triplet loss.

E. Training & Inference Speed

The system used for all speed tests used a Geforce GTX 970 graphics card, a AMD Ryzen 5 3600X processor and 16 GB RAM. The training and inference speeds for different MIRS models were nearly identical. The same is true for the different baselines models. We therefore only compare these two categories with each other.
It has to be noted that the training times given for the MIRS models is referring to the MIRS models using a DenseNet trained on ImageNet as a feature extractor which is publicly available. For other feature extractors the training time for the feature extractor also has to be considered. The feature extractor type however does not significantly affect the query execution time or the archive size.

As evident in Table IX, the MIRS models offer greatly increased training time since the only network that has to be trained is a small three-layer hashing network. We further improve this aspect by calculating the DenseNet features for all training and validation images at the start of training and then keep them in memory for the remainder of the training.

TABLE IX
SPEED COMPARISONS BETWEEN BASELINE USING ONLY DENSENET AND MIRS SYSTEMS USING HASHING NETWORK.

	Baseline	MIRS model
Training time per epoch	174s	1s
Total training time for 100 epochs	1740s	159s+100s=259s
Query execution time per image	6s	3s
Archive size	120 Mb	6.7 Mb

Without this optimisation the DenseNet would still need to be evaluated in every epoch for every training image which would negate a large part of the performance benefits. By using this optimisation, we reach a 174x speedup in training time per epoch and overall an almost 7x speedup in total training time when training for 100 epochs. Since the optimisation only needs to be done once at the start the total speed boost also converges to 174x for more epochs.
The execution time for an image query also decreases with the usage of the hashing network. The reason for this is that the archive now contains compact binary hashes instead of floating-point valued feature vectors. Thus, less memory has to be searched and the hashing operations in general are much more time efficient than the Euclidean distance calculations in the baseline model. In fact the archive generated by MIRS was 12x smaller than the one produced by the baseline.
Image retrieval for our baseline is still fast, because the DoMars16k data set is so small that the produced archive

only has a size of 120 Mb which is still easily searchable. However, for larger data sets the difference in speed would rise significantly.
In remote sensing tasks, the data sets that are worked with can often be very large. For this reason the immense performance increases demonstrated by our MIRS system are especially important in this field of study.

V. Conclusion

In this paper, we introduce MIRS, a CBIR system for Mars images, that offers both a supervised and an unsupervised learning approach while integrating hashing and methods to improve its generalisation capability. We thoroughly evaluated possible MIRS configurations and compared the best versions with a baseline setup based on direct feature comparisons. Even without training a dedicated feature extractor, which is the most time consuming step of the training process, MIRS achieves high precision for image retrieval. Yet, our results suggest that CBIR systems greatly benefit from training the extractor. We combined hashing and DWT and found that DWT can improve the performance of our system. However, to validate to which extent DWT improves performance we would have to run further tests on different data sets (preferably consisting of Mars images). Because our model also offers an unsupervised learning approach, it can not only be applied to annotated data sets but also to unlabelled data and is therefore provides a high degree of flexibility. Our unsupervised approach yielded good results in terms of retrieval accuracy, but still considerably lacks behind the supervised MIRS. Nevertheless, it is a good alternative if labelled data is scarce. Furthermore, the results of supervised MIRS imply that our unsupervised approach could also notably profit from training the feature extractor. Due to hardware constraints we could however not test this hypothesis. Thus, further experimental investigations are needed. Finally, MIRS is not only more accurate than our baseline, but also considerably more efficient. It requires only a fraction of memory to store the archive, due to the similarity preserving hash codes that consume little memory space. The compressed information combined with the quick hashing operations lead to rapid retrieval speeds compared to the baseline. As a consequence, our system is more scalable and can also be applied to bigger data sets. In summary, MIRS is an accurate, flexible and scalable image retrieval system.

REFERENCES

[1] H. Hargitai and A. Naß, "Planetary mapping: A historical overview," in *Planetary Cartography and GIS*, ser. Lecture Notes in Geoinformation and Cartography, H. Hargitai, Ed. Springer International Publishing, 2019, pp. 27–64. [Online]. Available: https://doi.org/10.1007/978-3-319-62849-3_2

[2] G. Sumbul, J. Kang, and B. Demir, "Deep learning for image search and retrieval in large remote sensing archives," *ArXiv*, 2020. [Online]. Available: http://arxiv.org/abs/2004.01613

[3] I. Tekeste and B. Demir, "Advanced local binary patterns for remote sensing image retrieval," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 6855–6858.

[4] S. R. Dubey, S. K. Singh, and R. K. Singh, "Multichannel decoded local binary patterns for content-based image retrieval," *IEEE Transactions on Image Processing*, vol. 25, no. 9, pp. 4018–4032, 2016.

[5] Y. Yang and S. Newsam, "Geographic image retrieval using local invariant features," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 2, pp. 818–832, 2013.

[6] S. Roy, E. Sangineto, B. Demir, and N. Sebe, "Metric-learning-based deep hashing network for content-based retrieval of remote sensing images," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 2, pp. 226–230, 2021.

[7] G. Shakhnarovich, T. Darrell, and P. Indyk, "Nearest-neighbor methods in learning and vision: theory and practice," ser. Neural information processing series, G. Shakhnarovich, T. Darrell, and P. Indyk, Eds. MIT Press, 2005.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: http://www.nature.com/articles/nature14539

[9] C. Yan, B. Gong, Y. Wei, and Y. Gao, "Deep multi-view enhancement hashing for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 4, pp. 1445–1451, 2021, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[10] S. Roy, A. Siarohin, E. Sangineto, S. R. Bulo, N. Sebe, and E. Ricci, "Unsupervised domain adaptation using feature-whitening and consensus loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] J. Pujari, S. Pushpalatha, and D. Padmashree, "Content-based image retrieval using color and shape descriptors," in *2010 International Conference on Signal and Image Processing*, 2010, pp. 239–242.

[12] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: the qbic system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[13] S. Brandt, J. Laaksonen, and E. Oja, "Statistical shape features in content-based image retrieval," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 2, 2000, pp. 1062–1065 vol.2.

[14] J. R. Smith and S.-F. Chang, "Visualseek: A fully automated content-based image query system," in *Proceedings of the Fourth ACM International Conference on Multimedia*, ser. MULTIMEDIA '96. New York, NY, USA: Association for Computing Machinery, 1997, p. 87–98. [Online]. Available: https://doi.org/10.1145/244130.244151

[15] A. Vijay and M. Bhattacharya, "Content-based medical image retrieval using the generic fourier descriptor with brightness," in *2009 Second International Conference on Machine Vision*, 2009, pp. 330–332.

[16] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[17] Y. Zhang, L. Zhang, and Q. Tian, "A prior-free weighting scheme for binary code ranking," *IEEE Transactions on Multimedia*, vol. 16, no. 4, pp. 1127–1139, 2014.

[18] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *AAAI Conference on Artificial Intelligence*, 2016.

[19] Y. Li, Y. Zhang, X. Huang, H. Zhu, and J. Ma, "Large-scale remote sensing image retrieval by deep hashing neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 950–965, 2018.

[20] W. Song, S. Li, and J. A. Benediktsson, "Deep hashing learning for visual and semantic retrieval of remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–12, 2020.

[21] R. Cao, Q. Zhang, J. Zhu, Q. Li, Q. Li, B. Liu, and G. Qiu, "Enhancing remote sensing image retrieval with triplet deep metric learning network," *International Journal of Remote Sensing*, vol. 41, no. 2, pp. 740–751, 2020. [Online]. Available: http://arxiv.org/abs/1902.05818

[22] W. Xiong, Z. Xiong, Y. Zhang, Y. Cui, and X. Gu, "A deep cross-modality hashing network for sar and optical remote sensing images retrieval," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 5284–5296, 2020.

[23] Y. Li, Y. Zhang, X. Huang, and J. Ma, "Learning source-invariant deep hashing convolutional neural networks for cross-source remote sensing image retrieval," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 11, pp. 6521–6536, 2018.

[24] X. Lu, Y. Chen, and X. Li, "Hierarchical recurrent neural hashing for image retrieval with hierarchical convolutional features," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 106–120, 2018.

[25] P. Li, L. Han, X. Tao, X. Zhang, C. Grecos, A. Plaza, and P. Ren, "Hashing nets for hashing: A quantized deep learning to hash framework for remote sensing image retrieval," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 10, pp. 7331–7345, 2020.

[26] H. Zhang, L. Liu, Y. Long, and L. Shao, "Unsupervised deep hashing with pseudo labels for scalable image retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1626–1638, 2018.

[27] S. Roy, E. Sangineto, B. Demir, and N. Sebe, "Deep metric and hash-code learning for content-based retrieval of remote sensing images," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 4539–4542.

[28] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682

[29] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information science and statistics. Springer, 2006.

[30] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv:2002.05709 [cs, stat]*, 2020. [Online]. Available: http://arxiv.org/abs/2002.05709

[31] T. Milbich, K. Roth, B. Brattoli, and B. Ommer, "Sharing matters for generalization in deep metric learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[32] A. Siarohin, E. Sangineto, and N. Sebe, "Whitening and coloring batch transform for GANs." [Online]. Available: http://arxiv.org/abs/1806.00420

[33] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, "Deep learning for content-based image retrieval: A comprehensive study," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 157–166. [Online]. Available: https://doi.org/10.1145/2647868.2654948

[34] T. Wilhelm, M. Geis, J. Püttschneider, T. Sievernich, T. Weber, K. Wohlfarth, and C. Wöhler, "Domars16k: A diverse dataset for weakly supervised geomorphologic analysis on mars," *Remote Sensing*, vol. 12, no. 23, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/23/3981

[35] E. D. McKee, "A study of global sand seas," vol. 1052, 1979, USGS Numbered Series, p. 439. [Online]. Available: http://pubs.er.usgs.gov/publication/pp1052