Grid-Based Concise Hash for Solar Images

Rafał Grycuk[1][0000−0002−3097−985X] and Rafał Scherer[1][0000−0001−9592−262X]

Czestochowa University of Technology, al. Armii Krajowej 36, Czestochowa, Poland
{rafal.grycuk,rafal.scherer}@pcz.pl

Abstract. Continuous full-disk observations of the solar chromosphere and corona are provided nowadays by the Solar Dynamics Observatory. Such data are crucial for analysing the Sun-Earth system and life on our planet. Part of the data is an enormous number of high-resolution images. We create a compact grid-based solar image hash to classify or retrieve similar solar images. To compute the hash, we design intermediate hand-crafted features. Then, we use a convolutional autoencoder to encode the descriptors to the form of a concise hash.

Keywords: Fast image hash · Solar activity analysis · Solar image description · CBIR of solar images.

1 Introduction

The NASA Solar Dynamics Observatory spacecraft has been providing solar data since 2010. Its part, the Atmospheric Imaging Assembly (AIA) delivers continuous full-disk observations of the solar chromosphere and corona in seven extreme ultraviolet (EUV) channels with the 12-second cadence in the form of high-resolution 4096×4096 pixel images. The images are relatively similar to each other, and general-purpose visual features are not suitable for their description. Moreover, the images are denoted only by their timestamp.

Semantic hashing [18] aims at generating compact vectors which values reflect semantic content of the objects. Thus, to retrieve similar objects we can search for similar hashes which is much faster and takes much less memory than operating directly on the objects. In [18] a multilayer neural network was used to generate hashes. Learned semantic hashes [20] are gaining in popularity in image retrieval. Our initial attempts showed that computing hashes from full-disk solar images would not be viable taking into account the size of the solar image collections (in terms of resolution and the number of images). Therefore, we developed the aforementioned hand-crafted intermediate descriptors.

A full-disk content-based image retrieval system is described in [1]. The authors checked eighteen image similarity measures with various image features resulting in one hundred and eighty combinations. The experiments shed light on what metrics are suitable for comparing solar images to retrieve or classify various phenomena.

A general-purpose retrieval engine Lucene is used to retrieve solar images in [2]. Each image is a document consisting of 64 elements (rows of each image), and

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

every image-document is unique. The solar images are then queried by setting some wild-card characters in the query strings that allows to search for similar solar events. The Lucene engine is compared in [3] with distance-based image retrieval methods, however, without a clear winner. It turned out that every tested method has its pros and cons in terms of accuracy, speed and applicability. The trade-off between accuracy and speed is significant, and for accurate results, the retrieval time was several minutes.

A sparse model representation of solar images was developed in [8]. The method used the sparse representation from [13] and outperformed previous solar image retrievals in accuracy and speed. In [10], some solar image parameters are chosen to track multiple solar events across images with 6-minute cadence. Sparse codes for AIA images are used also in [9], where ten texture-based image parameters are used to create the code. The parameters are computed for regions determined by a 64×64 grid for nine wavelengths. For each wavelength, a dictionary of k elements is learned, and then a sparse representation is computed. To overcome the curse of dimensionality affecting the solar data, they use the Minkowski norm and choose the right value of p parameter. Finally, the authors used a 256-dimensional descriptor what is an efficient and accurate outcome comparing to the previous approaches. In [11], a method for image retrieval with fuzzy sets and boosting is developed.

To automate solar image retrieval and enable their fast classification, we propose a fast and concise solar image hash generated from one-dimensional hand-crafted features by a fully convolutional autoencoder. The hash has only eleven real-valued elements, and the experiments showed that such compactness is sufficient to describe the images. In the dataset, the images are annotated only by their timestamp. It is very hard to make any meaning to data or explain the trained system [14]. We treat the timestamp as a measure of similarity. After training, our algorithm allows retrieving images by their visual similarity, regardless of the timestamp proximity. The paper is organized as follows. Section 2 introduces the method for generating learned solar hashes. Experiments on the SDO solar image collection are described in Section 3. Section 4 concludes the paper.

2 Grid-based Image Hash for Solar Image Retrieval

Solar images are relatively similar to each other, and general-purpose descriptors are usually not applicable in their retrieval. Therefore, we present a novel grid-based algorithm for the solar image hashing. The proposed hash can be used for image retrieval of solar images in large solar image datasets. The solar images were taken from the Solar Dynamics Observatory (SDO), where they are post-processed and published in the form of Web API by [12]. There are many resolutions available, and we use the high definition 2048×2048 images;thus creating image descriptors requires a significant amount of memory. In our experimental environment, we used GeForce RTX 2080 Ti 11GB GDDR6 graphics card, which allowed us to use 11 GB of memory. Initially, we tried to design

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

directly a full-disc autoencoder. Setting a higher mini-batch value caused an out-of-memory exception. Moreover, the learning time in this simulation took several days versus several minutes as in the presented approach. Therefore, we decided to apply some preprocessing stage (calculation of grid-based descriptor) and then use the autoencoder (see Sec. 2.3) to reduce the hand-crafted vectors to 11-element real-valued hashes without losing significant information about the active regions. The presented algorithm is composed of four main stages: active region detection, calculating solar image hand-crafted descriptors, encoding to hash, and retrieval.

2.1 Active Region Detection

In the first step, we need to obtain the Active Regions (AR). They are brighter regions of the solar images, and they are essential in detecting solar flares. ARs have various shapes, and they change due to the Sun's rotation movement. The presented method determines the positions and shapes of Active Regions. During this process, at first, we change image colour space from RGB to grayscale. Therefore we reduce the number of colour channels from three to one. As a result, every pixel of our grayscale solar image will have intensities values in the range [0..255]. In the next stage, we use the Gaussian blur in order to remove insignificant, small regions. Then, the pixel intensities are filtered by using the threshold th, provided as the algorithm parameter. Thus, the obtained image is properly preprocessed for the thresholding stage. Subsequently, every pixel intensity is compared with the provided threshold parameter th value. If the value is greater or equal, we determine that pixel is a part of the active region. The value th parameter was determined empirically to 180, and it was obtained for the given solar image dataset [12]. In the next step, we apply the morphological operations, namely erosion and dilation to the thresholded image, obtained in the previous step. The morphological operation erosion eradicates separated small objects (pixels). These objects can be referred as "islands". After this operation, only substantive (important) objects remain. The dilation operation, on the other hand, makes objects more visible; thus, it fills in small holes in objects. These two types of operations can enhance the important areas of the active regions. More informations about morphological operations can be found in [5][19]. The process of active region detection is described in the form of pseudo-code in Algorithm 1. Figure 1 presents an input image (left) and active regions detected in the image (right). The applied operations allow detecting active regions of the solar image. The accurate detection of these regions is vital to subsequent stages of the algorithm. The location and the shape of active regions are significant in detecting the Coronal Mass Ejection (CME) and thus, in the solar flare prediction.

2.2 Calculating Grid-based Descriptor

In this section, we describe the process of calculating the grid-based descriptor. We take an image with active region detected (AR image) as input, and we obtain

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

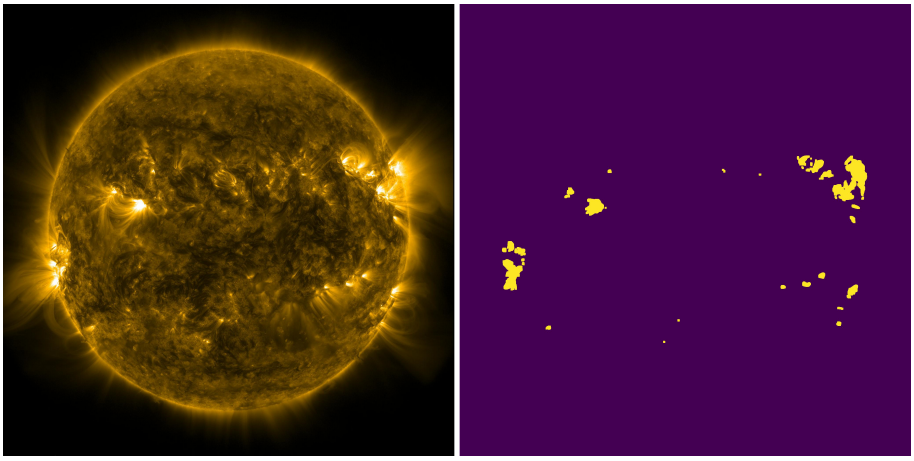

Fig. 1. Active region detection process, left image is the input, right image is the output.

INPUT: $SolarImage$
OUTPUT: $ActiveRegionDetectedImg$
$GrayScaleImg := ConvertGrayScale(SolarImage)$
$BlurredImg := Blur(GrayScaleImg)$
$ThreshImg := Threshold(BlurredImg)$
$ErodedImg := Erode(ThreshImg)$
$ActiveRegionDetectedImg = Dilate(ErodedImg)$

Algorithm 1: Active region detection steps.

a grid-based descriptor of 100 length. The descriptor length was determined empirically, but this can be changed by adjusting the parameters $gridSizeN$ and $gridSizeM$. Values of these variables are equal 10 for both of them, which gives us a 100-element descriptor vector. In the first step, we take the AR image and divide it into cells (sub-images) using the grid. During this process, we slice image at x-axis and afterwards for each slice (cells) we perform slicing at y-axis. Therefore, we obtain an image grid, where grid cells contain sub-images. By using the parameters ($gridSizeN$ and $gridSizeM$) we can define a number of grid cells both for x and y axis. In the next step, we calculate a sum of active region pixels for each grid cell. As a result of this process, we obtain $n \times m$ **DM** matrix, where each element contains a grid cell sum. In the last step of this stage, we perform a matrix normalization and vectorization and provide a **DV** vector of size $n * m$, e.g. if matrix **DM** is 10×10 then **DV** size is 100. It should be noted that **DV** size depends on values of $gridSizeN$ and $gridSizeM$ parameters. It also should be noted that both grid size parameters was obtained empirically, and there values have significant impact on the results. Scaling the values for both axis allows to determine the most suitable values for given solar image resolution. The entire process of calculating the grid-based descriptor is

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

presented in the form of pseudo-code in Alg. 2. Let us analyze the consecutive

INPUT: ARI - active region detected image
$gridSizeN$ - grid size in x-axis
$gridSizeM$ - grid size in y-axis
OUTPUT: $GridBasedDescriptorVector$
Local Variables: $ImageCells$ - list for containing grid image cells
$SumMatrix$ - matrix sums of pixels in the cells
$HSlices := DivideIntoHorizontalSlices(ARI, gridSizeN)$
foreach $HSlice \in HSlices$ **do**
  $CellsForHSlice := DivideSlicesVerticallyIntoCells(HSlice, gridSizeM)$
  foreach $CellForHSlice \in CellsForHSlice$ **do**
    $ImageCells.Add(CellForHSlice)$
  end
end
foreach $ImageCell \in ImageCells$ **do**
  $CellSum := CalculateSumOfActiveRegionPixelsInCell(ImageCell)$
  $SumMatrix.SetCellSum(CellSum)$
end
$GridBasedDescriptorVector = VectorizeMatrix(SumMatrix)$

Algorithm 2: Algorithm for calculating grid-based descriptor.

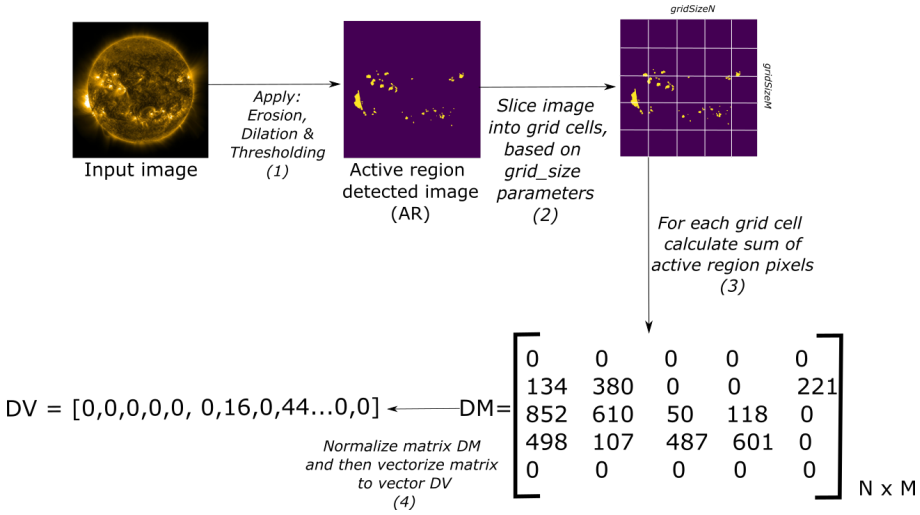

Fig. 2. Steps for calculation of the grid-based descriptor.

steps of this stage in the visual form; see Fig. 2. In the first step, the input image is subjected to morphological operations of erosion and dilation, and then the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

thresholding process is applied. As a result, we obtain the active region detected image. This process is defined by Eq. 1. In the next step, we slice the image into the grid cells. Based on the previously obtained grid, we calculate **DM** matrix. This stage can be performed by using Eq. 2. Afterwards, we normalize this matrix and then vectorize it in order to obtain the **DV** vector

$$t(ARI, i, j, th) = \begin{cases} 1, & ARI_{i,j} \geq th \\ 0, & \text{otherwise} \end{cases}, \tag{1}$$

where th is threshold value and ARI is active region image.

$$\mathbf{DM}(ARI, th)_{k,l} = \sum_{i=k*csx}^{(k+1)*csx-1} \sum_{j=l*csy}^{(l+1)*csy-1} t(ARI, i, j, th) \,, \tag{2}$$

where csx is the cell size in x-axis and csy is the cell size in y-axis. The process of calculating grid-based descriptor allows reducing the data volume during the encoding stage significantly. The aim of this process is to obtain a hand-crafted, intermediate mathematical representation of AR images used in the next step.

2.3 Hash Generation

In this section, the hash generation (autoencoding) process is described. We use previously obtained grid-based image descriptors to reduce the descriptor length and in order to obtain the latent space a one-dimensional hash. To this task, we used a convolutional autoencoder to encode our hand-crafted image descriptors in its latent space. Autoencoders are used for network user identification [6, 16] or for image reconstruction and improvement [15, 17]. We used the unsupervised convolutional neural network as it does not require labelled data for training (it was not provided in the Web API or the dataset). The autoencoder architecture is presented in Tab. 1. As can be seen in Table 1, a convolutional autoencoder was used for hash generation process, and the table should be analyzed top to bottom, where the top layer is input. Afterwards, we have a set of encoding layers; it is composed of 3-layer groups, where every group contains three layers ($Conv1D$, $ReLU$, $MaxPool1D$). The *kernel* parameter used in $Conv1D$ and $MaxPool1D$ layers is equal 2. After three convolutional layers with pooling, we have the latent space, bottleneck layer, which is encoded layer for the hash generation. Then, the autoencoder has decoding groups which are composed of $MaxUnPool1D$ (upsampling) layers, convolutional layers and $ReLU$. There is also a padding layer, which allows obtaining the same shape of decoded data as the input one. For the hash generation, we only use the encoding layers. The reason why we used a one-dimensional autoencoder is that our grid-based image descriptors are one-dimensional vectors for decreasing computational complexity. This process allows reducing the hash length without significant loss of important information about the active regions of the solar image. For the loss function, we used the mean squared error function. We empirically proved that 50 epochs are sufficient to obtain the required level of generalization and to prevent the network over-fitting.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

Table 1. Tabular representation of the convolutional autoencoder model.

Layer (type)	Output Shape	Filters (in, out)	Kernel size	Params no.
$Input1d(InputLayer)$	[1, 1, 100]			
$Conv1d\_1(Conv1D)$	[1, 64, 100]	1, 64	2	192
$ReLU\_1$	[1, 64, 100]			
$Max\_pooling1d\_1(MaxPool1D)$	[2, 64, 48]		2	
$Conv1d\_2(Conv1D)$	[2, 32, 48]	64, 32	2	4128
$ReLU\_2$	[2, 32, 48]			
$Max\_pooling1d\_2(MaxPool1D)$	[2, 32, 24]		2	
$Conv1d\_3(Conv1D)$	[1, 1, 23]	32, 1	2	65
$ReLU\_3$	[1, 1, 11]			
$Encoded(MaxPool1D)$	[1, 1, 11]		2	
$up\_sampling1d\_1(MaxUnPool1D)$	[1, 1, 22]		2	
$ConvTranspose1d\_1(ConvTranspose1D)$	[1, 1, 23]	1, 32	2	96
$ReLU\_4$	[1, 1, 23]			
$ConstPadding_1(ConstPad1D)$	[1, 1, 24]		2	
$up\_sampling1d\_2(MaxUnPool1D)$	[2, 32, 48]		2	
$ConvTranspose1d\_2(ConvTranspose1D)$	[2, 64, 49]	32, 64	2	4160
$ReLU\_5$	[2, 64, 49]			
$up\_sampling1d\_3(MaxUnPool1D)$	[1, 64, 98]		2	
$ConvTranspose1d\_3(ConvTranspose1D)$	[1, 1, 99]	61, 1	2	129
$ReLU\_6$	[1, 1, 99]			
$ConstPadding_1(ConstPad1D)$	[1, 1, 100]		2	
$Decoded(Tanh)$	[1, 1, 100]			

After the training process is finished, every image descriptor is provided to the latent space (encoded) layers of the autoencoder. As a result of this process, we obtained encoded a fast image hash as a 11-tuple of real-value elements. As can be seen in Table 2, the presented method provides the image hashes, where hashes of consecutive images are similar, which is highly desirable, because consecutive solar corona images have similar active regions. The obtained hash can be used for content-based solar image retrieval applications. It also should be noted that presented autoencoder architecture was selected in order to obtain the most suited generalization level.

2.4 Retrieval

In the last stage of the presented method, we use previously obtained hashes for image retrieval. After previous steps, we can assume that every solar image has a hash assigned in our image database. The retrieval step allows executing the image query by comparing distances between the query image hash and hashes created for all images stored in the dataset. The retrieval step requires to have a solar image database with a hash generated for every image. In the next step, we calculate the distances between the query image hash and every hash in the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

Table 2. Two examples of similar image pairs with their corresponding 11-element hashes. They show how similar are the vectors for semantically similar images.

Hash values			
Pair 1		Pair 2	
2015-01-01 00:00:00	2015-01-01 00:06:00	2015-03-02 04:06:00	2015-03-02 04:12:00
0.18948224	0.18948224	0.09669617	0.09669617
0.38965224	0.34482240	0.09669617	0.09669617
0.18948224	0.18948224	0.09669617	0.09669617
0.18948224	0.18948781	0.10213041	0.10213816
0.18943328	0.18936896	0.10843775	0.10845160
0.18830273	0.18814683	0.11238195	0.11103466
0.18947415	0.18947072	0.10204165	0.10199506
0.19400954	0.19399905	0.11846152	0.11823325
0.18971351	0.18972327	0.10581696	0.10593924
0.18960209	0.18960896	0.09670250	0.09670459
0.18948224	0.18948224	0.09669617	0.09669617

database. The distance d is calculated by the cosine distance measure (for more see [7])

$$\cos(QH_j, IH_j) = \sum_{j=0}^{n} \frac{(QH_j \bullet IH_j)}{\|QH_j\| \, \|IH_j\|}, \tag{3}$$

where $\bullet$ is dot product, QH_j is the query image hash, and IH_j a consecutive image hash. After calculating the cosine distance, the images stored in the database are sorted in ascending order by distance to the query (query hash). The last step of the presented method allows to take n images closest to the query and return them to the user as the retrieved images. During query execution, the n parameter is required. The entire process is presented as pseudo-code in Alg. 3. Alternatively, we can also retrieve images based on a threshold. In such a case, we must provide a threshold parameter instead of n and then retrieve images only if their cosine distance to the query is below the threshold. The proposed method also allows applying such an approach. Nevertheless, we prefer the first method because it is more suited for the system user.

INPUT: $ImageHashes$, $QueryImage$, n
OUTPUT: $RetrievedImages$
foreach $ImageHash \in ImageHashes$ **do**
  $QueryImageHash = CalculateHash(QueryImage)$
  $D[i] = Cos(QueryImageHash, ImageHash)$
end
$SortedDistances = SortAscending(D)$
$RetrievedImages = TakeFirst(n)$

Algorithm 3: Image retrieval steps.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

3 Experimental Results

This section describes simulation results along with a solution for the evaluation of unlabelled images. Due to lack of labelled data, unsupervised learning was used for descriptors encoding. Therefore, the evaluation of the proposed method with state of the art approaches is difficult. In order to resolve this problem, we use the Sun's rotation movement to determine a set of similar images (SI). We assumed that consecutive images within a small time window should have similar active regions. Those regions are slightly shifted between consecutive images. The Web API provides solar images with 6-minute cadence window. Due to the nature of the Sun movement, we can assume the similarity of consecutive images. The only condition is adjusting the difference time window. Based on experiments, we determined that images within a 48-hour window can be treated as similar. Let us take under consideration an image taken at 2012-02-15, 00:00:00. Based on the above assumptions, we can assume that every image in 24 hours before and in 24 hours after is similar. Only for evaluation purposes, images are identified by the timestamps. The process of determining similar images is presented in Table 3. By using the proposed method for determining image similarity we

Table 3. Defining image similarity. Based on experiments, we determined that images within a 48-hour window can be treated as similar. This allows to evaluate the method.

Timestamp	SI (similar image)/ NSI (not similar image)
2012-02-13, 23:54:00	NSI
2012-02-14, 00:00:00	SI
2012-02-14, 00:06:00	SI
2012-02-14, 00:12:00	SI
2012-02-14, 00:18:00	SI
2012-02-14, 00:24:00	SI
2012-02-14, 00:30:00	SI
.....	SI
2012-02-15, 00:00:00	QI (query image)
.....	SI
2012-02-15, 23:24:00	SI
2012-02-15, 23:30:00	SI
2012-02-15, 23:36:00	SI
2012-02-15, 23:42:00	SI
2012-02-15, 23:48:00	SI
2012-02-15, 23:54:00	SI
2012-02-16, 00:00:00	NSI

performed series of experiments and we obtained the similar images (SI).The single experiment can be described by the following steps:

1. Execute image query and obtain the retrieved images.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

2. For every retrieved image, compare its timestamp with the query image timestamp.
3. If the timestamp is the 48 hour window, the image is similar to the query.

After defining similar images (SI), we can define performance measures *precision* and *recall* [4][21] based on following sets:

- SI - set of similar images,
- RI - set of retrieved images for query,
- $PRI(TP)$ - set of positive retrieved images (true positive),
- $FPRI(FP)$ - false positive retrieved images (false positive),
- $PNRI(FN)$ - positive, not retrieved images,
- $FNRI(TN)$ - false, not retrieved images (TN).

Afterwards, we can define *precision* and *recall* for CBIR systems

$$precision = \frac{|PRI|}{|PRI + FPRI|}, \tag{4}$$

$$recall = \frac{|PRI|}{|PRI + PNRI|}. \tag{5}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall}. \tag{6}$$

Timestamp	RI	SI	PRI (TP)	FPRI (FP)	PNRI (FN)	Precision	Recall	F_1
2015-01-01 00:00:00	164	241	159	5	82	0.97	0.66	0,78
2015-01-03 01:00:00	372	481	330	42	151	0.89	0.69	0,77
2015-01-09 16:00:00	372	481	336	36	145	0.90	0.7	0,79
...	...	...	...	...	...	...	...	...
2015-05-12 00:36:00	362	481	330	32	151	0.91	0.69	0,78
2015-05-18 07:36:00	337	481	331	6	150	0.98	0.69	0,81
2015-05-25 18:36:00	349	481	317	32	164	0.91	0.66	0,76
...	...	...	...	...	...	...	...	...
2015-08-09 13:18:00	344	481	305	39	176	0.89	0.63	0,74
2015-08-11 05:24:00	327	481	315	12	166	0.96	0.65	0,77

Table 4. Experiment results for the proposed algorithm, performed on AIA images obtained from [12]. Due to lack of space, we present only a part of all queries.

We present the experiment results in Tab. 4. The presented results proved the effectiveness of the method. Our approach obtains a high value of the *precision* measure. Most of the images close to the query are correctly retrieved. The

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

farther from the query then more positive, not retrieved images (PNRI) are retrieved. This phenomenon is caused by the Sun's rotation, and thus more missing active regions are detected between images. In the 48-hour cadence, the significant active region can change its position; this may have a significant impact on the hash. Therefore, the distance to the query will be increased. The simulation environment was created in Python using Pytorch on the following hardware: Intel Core I9-9900k 3.6 GHz, 32 GB RAM, GeForce RTX 2080 Ti 11 GB, Windows Server 2016. The presented solution is available on the BitBucket repository under the following link: https://bitbucket.org/rafalgrycuk/novel_grid-based_image_hash_for_ content/src/src/master. The hash creation time took approximately 17.6 minutes, for 83, 819 images. The encoding stage took approximately 1.5 hours. The average retrieval time is approximately 300 ms.

4 Conclusions

In this paper, we proposed a novel grid-based image hash for fast content-based solar image retrieval and classification. Initially, we tried to make hashes directly from full-disc images. It turned out to be infeasible having general-purpose GPUs at our disposal. For this reason, we decided to design intermediate hand-crafted features. To this end, we apply morphological operations for preprocessing and active regions detection and then the grid for descriptor calculation. Only after this step, we use an unsupervised convolutional autoencoder to encode the descriptors to the concise hash form. The process of the second encoding allows reducing the description length significantly; in our experiments, over ten times compared to the hand-crafted descriptor obtained in the first stage. Reducing the hash length is, of course, significant for the speed of calculating the distances between hashes, that is, the similarity of solar images. As solar AIA images are unlabelled, we treat images generated in a short time to each other (up to several hours) as similar. In fact, at other time, the Sun configuration could be similar. Therefore, our precision and recall measures which rely on the image content solely will have even higher values in practice. The presented approach has various potential applications. It can be used for searching, classifying and retrieving solar flares, which has crucial importance for many aspects of life on Earth.

References

1. Banda, J., Angryk, R., Martens, P.: Steps toward a large-scale solar image data analysis to differentiate solar phenomena. Solar Physics **288**(1), 435–462 (2013)
2. Banda, J.M., Angryk, R.A.: Scalable solar image retrieval with lucene. In: 2014 IEEE International Conference on Big Data (Big Data). pp. 11–17. IEEE (2014)
3. Banda, J.M., Angryk, R.A.: Regional content-based image retrieval for solar images: Traditional versus modern methods. Astronomy and computing **13**, 108–116 (2015)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

4. Buckland, M., Gey, F.: The relationship between recall and precision. Journal of the American society for information science **45**(1), 12 (1994)
5. Dougherty, E.R.: An introduction to morphological image processing. SPIE, 1992 (1992)
6. Gabryel, M., Grzanek, K., Hayashi, Y.: Browser fingerprint coding methods increasing the effectiveness of user identification in the web traffic. Journal of Artificial Intelligence and Soft Computing Research **10**(4), 243–253 (2020). https://doi.org/10.2478/jaiscr-2020-0016
7. Kavitha, K., Rao, B.T.: Evaluation of distance measures for feature based image registration using alexnet. arXiv preprint arXiv:1907.12921 (2019)
8. Kempoton, D., Schuh, M., Angryk, R.: Towards using sparse coding in appearance models for solar event tracking. In: 2016 19th International Conference on Information Fusion (FUSION). pp. 1252–1259 (2016)
9. Kempton, D.J., Schuh, M.A., Angryk, R.A.: Describing solar images with sparse coding for similarity search. In: 2016 IEEE International Conference on Big Data (Big Data). pp. 3168–3176. IEEE (2016)
10. Kempton, D.J., Schuh, M.A., Angryk, R.A.: Tracking solar phenomena from the sdo. The Astrophysical Journal **869**(1), 54 (2018)
11. Korytkowski, M., Senkerik, R., Scherer, M.M., Angryk, R.A., Kordos, M., Siwocha, A.: Efficient image retrieval by fuzzy rules from boosting and metaheuristic. Journal of Artificial Intelligence and Soft Computing Research **10**(1), 57–69 (2020). https://doi.org/10.2478/jaiscr-2020-0005
12. Kucuk, A., Banda, J.M., Angryk, R.A.: A large-scale solar dynamics observatory image dataset for computer vision applications. Scientific data **4**, 170096 (2017)
13. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online learning for matrix factorization and sparse coding. Journal of Machine Learning Research **11**(Jan), 19–60 (2010)
14. Mikołajczyk, A., Grochowski, M., Kwasigroch, A.: Towards explainable classifiers using the counterfactual approach - global explanations for discovering bias in data. Journal of Artificial Intelligence and Soft Computing Research **11**(1), 51–67 (2021). https://doi.org/10.2478/jaiscr-2021-0004
15. Najgebauer, P., Scherer, R., Rutkowski, L.: Fully convolutional network for removing dct artefacts from images. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207249
16. Nowak, J., Holotyak, T., Korytkowski, M., Scherer, R., Voloshynovskiy, S.: Fingerprinting of url logs: Continuous user authentication from behavioural patterns. In: Krzhizhanovskaya, V.V., Závodszky, G., Lees, M.H., Dongarra, J.J., Sloot, P.M.A., Brissos, S., Teixeira, J. (eds.) Computational Science – ICCS 2020. pp. 184–195. Springer International Publishing, Cham (2020)
17. Pawlak, M., Panesar, G.S., Korytkowski, M.: A novel method for invariant image reconstruction. Journal of Artificial Intelligence and Soft Computing Research **11**(1), 69–80 (2021). https://doi.org/10.2478/jaiscr-2021-0005
18. Salakhutdinov, R., Hinton, G.: Semantic hashing. International Journal of Approximate Reasoning **50**(7), 969 – 978 (2009). https://doi.org/https://doi.org/10.1016/j.ijar.2008.11.006, special Section on Graphical Models and Information Retrieval
19. Serra, J.: Image analysis and mathematical morphology. Academic Press, Inc. (1983)
20. de Souza, G.B., da Silva Santos, D.F., Pires, R.G., Marananil, A.N., Papa, J.P.: Deep features extraction for robust fingerprint spoofing attack detection. Jour-

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20

nal of Artificial Intelligence and Soft Computing Research **9**(1), 41–49 (2019). https://doi.org/10.2478/jaiscr-2018-0023
21. Ting, K.M.: Precision and recall. In: Encyclopedia of machine learning, pp. 781–781. Springer (2011)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_20