Novel Fast Binary Hash for Content-based Solar Image Retrieval

Rafał Grycuk, Rafał Scherer
Department of Computational Intelligence
Czestochowa University of Technology
Al. Armii Krajowej 36, 42-200 Czestochowa, Poland
rafal.grycuk@pcz.pl, rafal.scherer@pcz.pl

***Abstract*—The Solar Dynamics Observatory provides data to research the connected Sun-Earth system and the impact of the Sun on living on the Earth. Its part, the Atmospheric Imaging Assembly, performs continuous full-disk observations of the solar chromosphere and corona in seven extreme ultraviolet channels with the 12-second cadence of high-resolution, over 16-megapixel images. In the paper, we create a fast binary hash to retrieve similar solar images in this vast collection. We use a fully convolutional autoencoder working on preprocessed solar full-disk projections.**

***Index Terms*—solar activity analysis, solar image description, CBIR, fast binary hashes**

I. Introduction

In 2010 NASA launched the Solar Dynamics Observatory (SDO) as a part of the Living with a Star program. The goal was to provide data to research the connected Sun-Earth system and the impact of the Sun on living on the Earth. The Sun activity, for example, huge electromagnetic storms, can affect electronics, navigation systems, or electric power grids. The SDO is a 3-axis stabilized spacecraft with three main sensory instruments. One of them is the Atmospheric Imaging Assembly (AIA), which provides continuous full-disk observations of the solar chromosphere and corona in seven extreme ultraviolet (EUV) channels with the 12-second cadence of high-resolution 4096×4096 pixel images. The commencement of the SDO program allowed to analyze the solar activity, yet, with all the problems associated with big data, as SDO generates about 70 thousand images a day. It is impossible to search and annotate manually such a waste collection of images. Moreover, this type of images is very repetitive and monotonous for humans, making the process even more troublesome.

In the paper, we create a fast binary hash to retrieve similar solar images in this vast collection. We use a fully convolutional autoencoder working on preprocessed solar full-disk projections. We focus on the efficiency of the retrieval process and the proposed method is faster than the existing methods with similar accuracy. The speed comes from the size of the proposed solar hashes. It is worth to mention that the direct full-disk solar image hashing is too computationally demanding what was the main rationale behind our work. The presented hash can be used in content-based image retrieval (CBIR) [1], [2] in solar domain.

The project financed under the program of the Polish Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019–2022 project number 020/RID/2018/19, the amount of financing 12,000,000.00 PLN.

The paper is organized as follows. Section II mentions some works on solar image retrieval. Section III introduces the method for generating learned solar hashes. Experiments on the SDO solar image collection are described in Section IV. Section V concludes the paper.

II. Related Works

A full-disk content-based image retrieval system is described in [3]. 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.

In [4] the full-disk SDO images images are segmented by the 64×64 grid into sub-images. Then, ten parameters, namely, entropy, fractal dimension, the mean intensity, the third and fourth moments, relative smoothness, the standard deviation of the intensity, Tamura contrast, Tamura directionality and uniformity are computed for each subimage.

A general-purpose retrieval engine Lucene is used to retrieve solar images in [5]. Each image is a document consisting of 64 elements (rows of each image), and 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 [6] 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 [7]. The method used the sparse representation from [8] and outperformed previous solar image retrievals in accuracy and speed. In [9] 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 [10], 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. Images from AIA can be also retrieved by a sequential analysis [11], [12].

Last years brought learned semantic hashes [13] to image retrieval. Semantic hashing [14] 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 then operating directly on the objects. The term was introduced in [14]. They used for the first time a multilayer neural network [15] to generate hashes. Autoencoder neural networks can be used also for image reconstruction [16], [17] when we do not use the latent space. Computing hashes from full-disk hi-res solar images would be not be viable taking into account the size of the solar image collections. That is way we create a novel fast binary hash from the proposed engineered features called further in the paper intermediate descriptors.

III. Proposed Fast Binary Hash for CBIR on Solar Images

In this section, we propose a novel approach for the solar image description. The proposed hash can be used for image retrieval of solar images in large repositories. We used images from the Solar Dynamics Observatory, which were later extracted and published in the form of Web API by [18]. Among many resolutions, we decided to use 2048×2048. We can distinguish three main stages in the hash creation: calculating binary descriptor, encoding, and retrieval.

A. Calculating Binary Descriptor

In this step, we have a 2048×2048 solar input image on which we can distinguish Active Regions (AR, see. Fig. 1), which are interesting in reference to solar flares. As can be seen, these regions can have various shapes, and they can change due to the Earth's rotation. In the first step of the presented method, we determine the positions and shapes of those active regions. This step is performed by filtering the pixel values higher than the provided threshold th. During this process, we first convert the input image to the grayscale, where pixel intensities have the following scale [0..255]. Afterward, we use the Gaussian blur in order to remove insignificant, small regions. Thanks to this process, we can analyze only important image features without noise. After those simple steps, our image is properly preprocessed for the thresholding stage. In this step, we compare every pixel intensity with the provided threshold th value. If the value is greater or equal, we assume that the pixel is a part of the active region pixel. The th value was obtained empirically, and it equals 180, and it was adjusted for the given type of solar images. Every pixel above this value is treated as an active region. Afterward, the thresholded image is subjected to morphological operations like erosion and dilation. The morphological erosion removes "islands" and small objects so that only substantive objects remain. The dilation makes objects more visible, fills in small holes in objects. Applying those two operations emphasize the important features of the active regions. More about morphological operations can be found in [19], [20]. The previously described process can be described in the form of pseudo-code Alg.1.

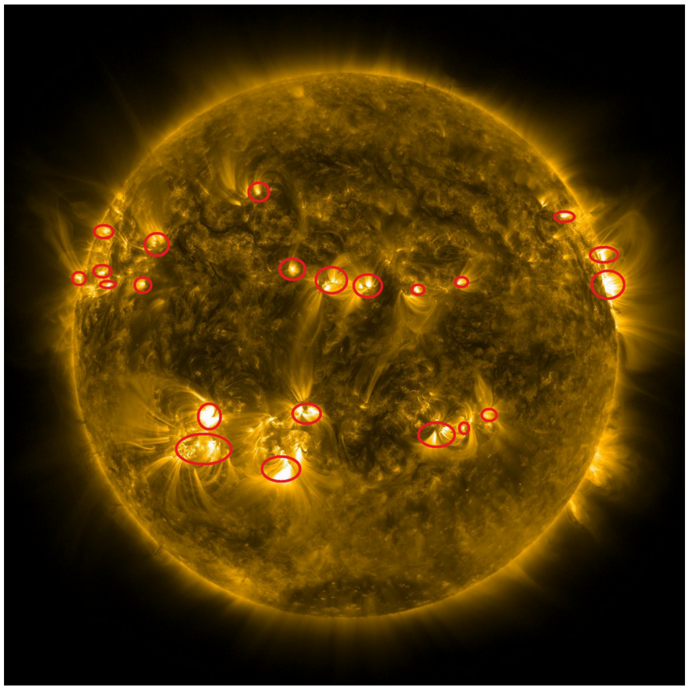

Fig. 1. Manually annotated solar active regions.

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

Algorithm 1: Active region detection steps.

The output image of the active region detection is presented in Fig. 2. As can be seen in Fig. 2, the applied operations allow to detect active regions. The location and shape of these regions are very important on account of the Coronal Mass Ejection (CME) and, thus, on the solar flare prediction.

After the detection of active regions, we need to create their representation in the form of a descriptor. The data set that we used provides images with 6-minute cadence; thus, we assumed that active regions change slightly between consecutive images. This phenomenon is caused by Earth's rotation. We propose an fast binary hash which is resistant to small changes of perspective. The consecutive image descriptors will have similar values. After the active region detection process, the descriptor calculation procedure is performed. In the first step,

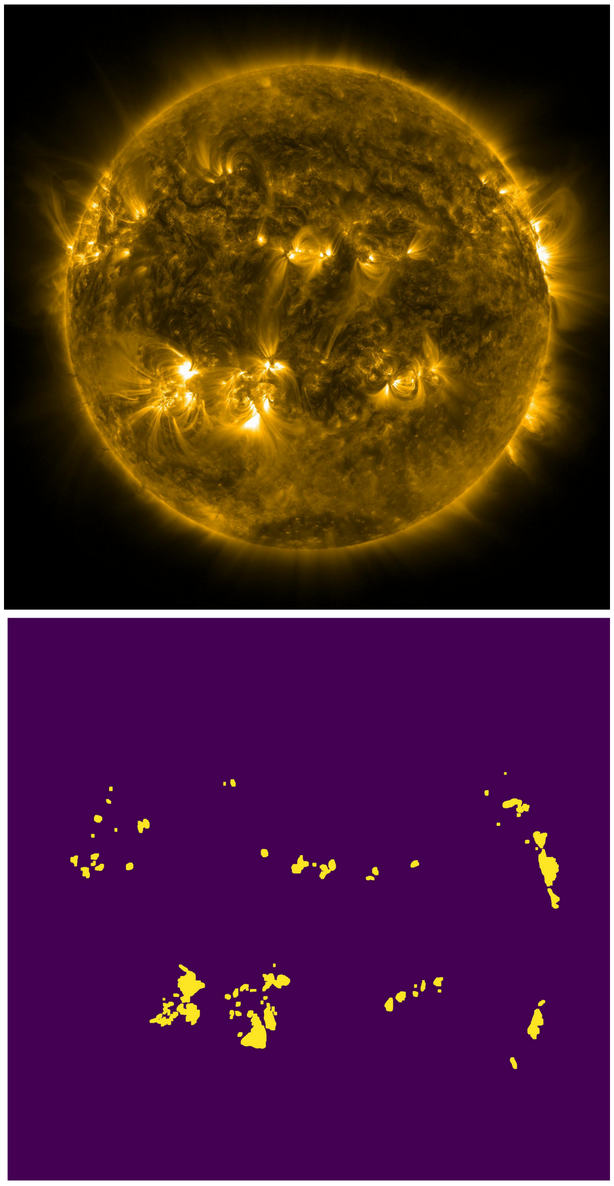

Fig. 2. Active region detection, top image is the input, bottom image is the output.

we need to calculate values of width vector $\mathbf{W}_j$ and height vector $\mathbf{H}_i$. Vectors $\mathbf{A}$ and $\mathbf{B}$ are auxiliary variables.

$$B_j = |\{i : ARI_{i,j} \geqslant th, i = 0, .., h\}|, j = 0, .., w$$

where w and h are width and height of the image active region, th is the threshold value.

$$\mathbf{W}_j = \sum_{i \in B_j} \mathbb{1}$$

Subsequently, we can define $\mathbf{H}_i$

$$A_i = |\{j : ARI_{i,j} \geqslant th, j = 0..., w\}|, i = 0, .., h$$

$$\mathbf{H}_i = \sum_{j \in A_i} \mathbb{1}$$

For every pixel in an active region we project it to height and width vectors $\mathbf{H}$, $\mathbf{W}$. This step is performed in order to obtain the combined $\mathbf{W}$ and $\mathbf{H}$ vectors as intermediate descriptors. This stage is presented in Fig. 3. In the last step we concatenate both $\mathbf{H}$ and $\mathbf{W}$ vectors. The projection process is also presented by the pseudo-code in Alg. 2. The

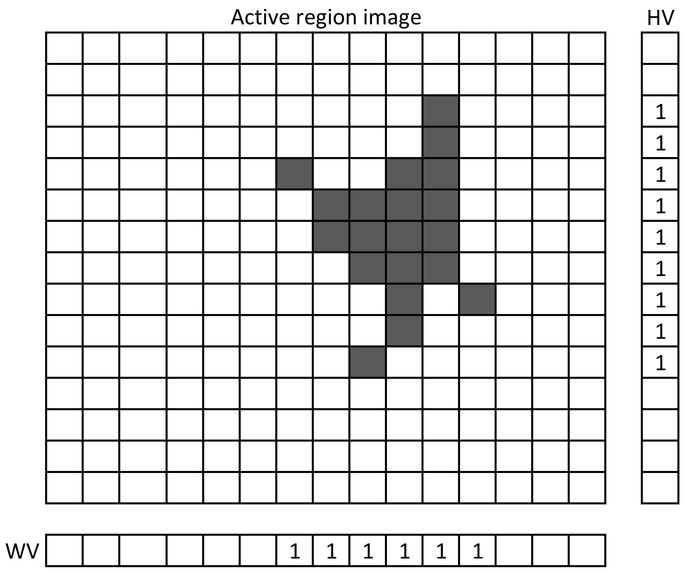

Fig. 3. Active region detection projection step.

```
INPUT: $ActiveRegionDetectedImg$, $th$
OUTPUT: $Descriptor$
foreach $i \in ActiveRegionDetectedImg$ do
    foreach $j \in ActiveRegionDetectedImg$ do
        if $ActiveRegionDetectedImg[i][j] \geqslant th$ then
            $\mathbf{H}[i] := 1$
            $\mathbf{W}[j] := 1$
        end
    end
end
$Descriptor := Concatenate(HV, WV)$
```

Algorithm 2: Active regions projection.

binary descriptor calculation stage allows reducing the amount of analyzed data significantly. The autoencoder applied in the encoding stage operates on one-dimensional data instead of two-dimensional, in the case of full-disc approaches. The presented approach allows reducing the time of data training, which will be significant when we add new images to the dataset. It should be noted that the computational complexity of the proposed method for descriptor calculation is $O(n^2)$. By using the projection of active regions, we reduce one dimension of our data. In the encoding step, we analyze data of length $\|\mathbf{H}\| + \|\mathbf{W}\|$ instead of $\|\mathbf{H}\| \cdot \|\mathbf{W}\|$, which conventional methods do.

B. Encoding

In this section the encoding process is described. We used a convolutional autoencoder for encoding in its latent space a one-dimensional hash from the previously obtained intermediate descriptors. The rationale behind the autoencoder is that it is a unsupervised convolutional neural network and does not require labelled data for training. The autoencoder architecture is presented in Fig. 4. A more detailed description

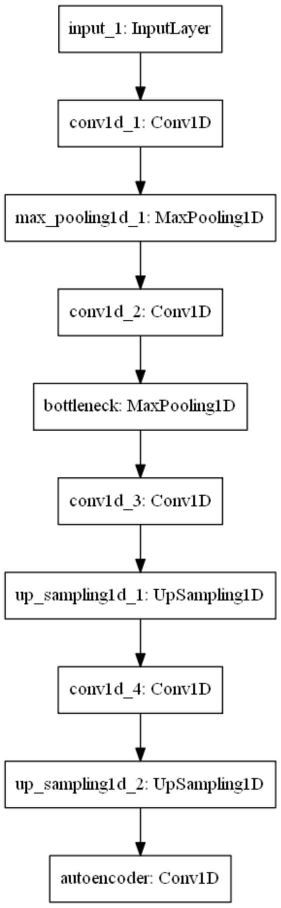

Fig. 4. Autoencoder model architecture. The bottleneck layer is the latent space for generating the fast binary hash.

of the model is presented in Tab. I. As can be seen in Fig. 4

TABLE I
TABULAR REPRESENTATION OF THE MODEL. THE SECOND COLUMN CONTAINS FEATURE MAP DIMENSIONS, NUMBER AND SIZE OF FILTERS.

Layer (type)	Output shape	Filters (no., size)
$input\_1(InputLayer)$	$(4096, 1)$	
$conv1d\_1(Conv1D)$	$(4096, 2)$	2,3
$max\_pooling1d\_1(MaxPool1D)$	$(2048, 2)$	
$conv1d\_2(Conv1D)$	$(2048, 1)$	1,3
$bottleneck(MaxPool1D)$	$(1024, 1)$	
$conv1d\_3(Conv1D)$	$(1024, 1)$	1,3
$up\_sampling1d\_1(UpSamp1D)$	$(2048, 1)$	
$conv1d\_4(Conv1D)$	$(2048, 2)$	2,3
$up\_sampling1d\_2(UpSamp1D)$	$(4096, 2)$	
$autoencoder(Conv1D)$	$(4096, 1)$	1,3

and Table I, we used an autoencoder with two convolutional layers with max-pooling, where $pool\_size$ parameter is equal 2. After two convolutional layers with pooling, we have the latent space, bottleneck layer, which is encoded layer. We used a one-dimensional autoencoder because we use it for the intermediate descriptor encoding. This stage allows us to reduce the hash length without significant loss of accuracy. By applying the encoding stage, we reduce the hash length for times over (4096 vs. 1024). After encoding layers, we have convolutional decoding layers with up-sampling. The decoding layers are used only for training. For a hash generation, we only use encoding layers. During the experiments, we used the Keras package along with Tensorflow 2.0. We applied the binary cross-entropy loss function. We empirically proved that 40 epochs are sufficient to obtain the required level of generalization and not cause the network over-fitting. The learning curve is presented in Fig. 5. After the learning process, every descriptor created in the previous step (Sec. III-A) is fed to the encoded layers of the autoencoder. As a result of this process, we obtained encoded fast binary hash of 1024 length. Such hash can be used for content-based solar image retrieval applications (see Sec. III-C).

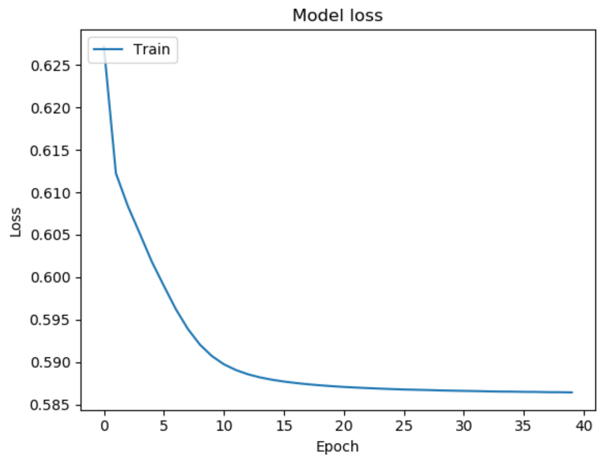

Fig. 5. The autoencoder learning curve.

C. Retrieval

In the last step of the presented method, we perform the image retrieval process. Let us assume that every solar image has a hash assigned in our image database. On the input of the image retrieval step, we have a query image; it is an image to which we will compare images stored in the database. In the first step of this process, we need to create the same type of hash that we created for all images in the database. The image retrieval scheme is presented in Fig. 6. As can be seen in Fig. 6, the encoded binary hash for the query image is generated. The retrieval step requires to have a solar image database with a hash generated for every image. In the next step, we calculate the distance between the query image hash and every hash in the database. The distance d is calculated by the cosine distance measure [21]

$$\cos(Q_j, I_j) = \sum_{j=0}^{n} \frac{(Q_j \bullet I_j)}{\|Q_j\| \, \|I_j\|},$$

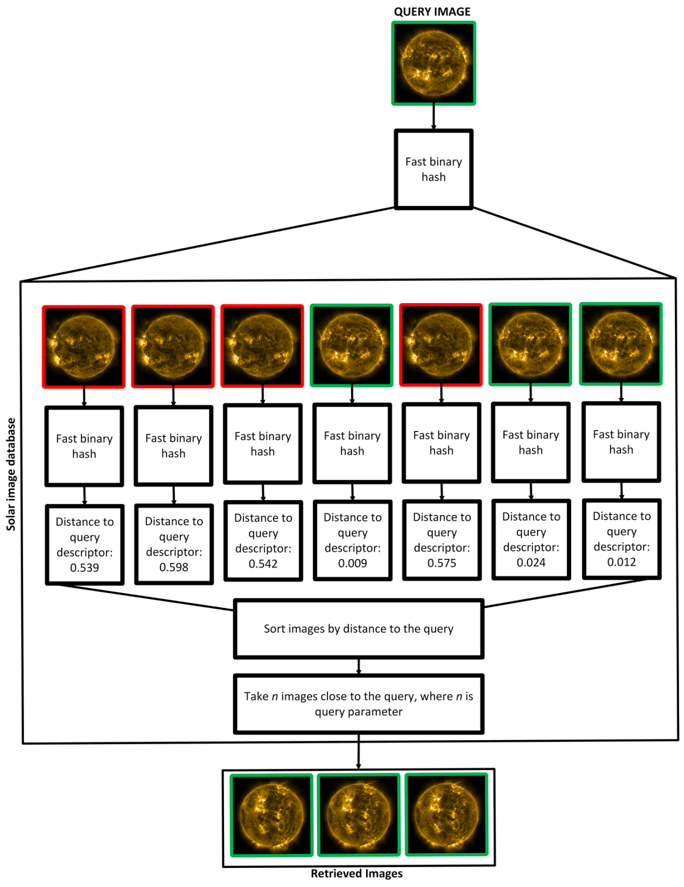
QUERY IMAGE
Fast binary hash
Solar image database
Fast binary hash
Distance to query descriptor: 0.539
Distance to query descriptor: 0.598
Distance to query descriptor: 0.542
Distance to query descriptor: 0.009
Distance to query descriptor: 0.575
Distance to query descriptor: 0.024
Distance to query descriptor: 0.012
Sort images by distance to the query
Take n images close to the query, where n is query parameter
Retrieved Images

Fig. 6. Active region detection projection step.

where • is dot dot product. During the experiments, we determined that this distance measure is the most suitable for the proposed hash. We also supported our decision by analyzing similar methods [21]. In the next step, the images

```
INPUT: $Hashes, QueryImage, n$
OUTPUT: $RetrivedImages$
foreach $hash \in Hashes$ do
    $QueryImageHash = CalculateHash(QueryImage)$
    $D[i] = Cos(QueryImageHash, hash)$
end
$SortedDistances = SortAscending(D)$
$RetrivedHashes = TakeFirst(n)$
$RetrievedImages =$
  $GetCorrespondingImages(RetrievedHashes)$
```
Algorithm 3: Image retrieval steps.

are sorted in the ascending order by the distance to the query. In the last step, we take n images closest to the query. Those images are returned to the user as the retrieved images. Parameter n is provided during query execution along with the query image. The entire retrieval process is described by Alg. 3.

IV. Experimental Results

In this section, we present experimental results along with our approach to method evaluation of unlabelled images. Due to the lack of labeled data, we had to resort to the unsupervised learning for encoding descriptors. Therefore, there is no possibility to evaluate the proposed method with state of the art approaches. We decided to develop a new approach to this problem. We can use Earth's rotation to determine similar images. The consecutive images within a small time window should have similar active regions. They should be slightly shifted relative to the previous image if we define the next and previous image by their creation date. The provided API allows fetching solar images with a 6-minute window (cadence). Therefore we can assume the similarity of consecutive images. The only condition is adjusting the difference time window. During performed experiments, we determined that a 48-hour window allows determining the similarity of the images. Let us take an image taken at 2012-02-15, 00:00:00; we can now assume that 24 hours before and 24 hours after the images are similar. Images are identified by the timestamps. This process is presented in Table II. Based

TABLE II
Defining image similarity.

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

on that assumption, we performed series of experiments in order to determine similar images (SI). The experiments are composed from the following steps:

1) Execute image query and obtain the retrieved images.
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) and (NSI) we can define performance measures $precision$ and $recall$ [22], [23]. They are defined using:

- 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).

Precision and *Recall* are defined as follows [23], [24]

$$prec = \frac{Total\ no.\ of\ retrieved\ relevant\ images}{Total\ number\ of\ retrieved\ images}, \quad (1)$$

$$recall = \frac{Total\ no.\ of\ retrieved\ relevant\ images}{Total\ number\ of\ relevant\ images}. \quad (2)$$

Based on the previously described sets and above formulas we can adapt formulas to CBIR needs

$$precision = \frac{|PRI|}{|PRI + FPRI|}, \quad (3)$$

$$recall = \frac{|PRI|}{|PRI + PNRI|}. \quad (4)$$

The performed experiment results were presented in Tables III and IV. As can be seen in the presented tables, our method obtains a high value of the *precision* measure. Most of the images close to the query are correctly retrieved. The farther from the query, the more positive not retrieved images (PNRI) are searched. This is caused by the Earth's rotation and missing active regions. If in the 48-hour window, the significant active region will change its position, this may significantly change the hash; therefore, the distance to the query will be increased. This is not a typical case, but it was observed during the experiments. The lower value of the *Recall* measure is caused by this phenomena. The presented simulation environment was created in Python using Tensorflow and Keras libraries. The presented solution is available on the BitBucket repository under the following link: https://bitbucket.org/rafalgrycuk/novel_encoded_solar_binary_descriptor/src/master.
The hash creation time was the longest stage; it took approximately 12 hours for 167638 images. The learning took 6 hours. The average retrieval time is 700 ms.

V. Conclusions

We proposed a novel fast binary hash for content-based solar image retrieval on solar images. The algorithm uses morphological operations for preprocessing and active regions detection, and then the binary descriptor is calculated. Afterward, we use an unsupervised convolutional autoencoder to encode the descriptors. After this process, we obtain the fast binary hash, which length was reduced four times over compared to the descriptor obtained before encoding. Reducing the hash length is significant relative to calculating the distances between hashes.The performed experiments and comparisons (see Tables III and IV) proved the efficiency of the proposed approach. The presented method has various potential applications. It can be used for searching and retrieving solar flares, which has crucial importance for many aspects of life on Earth.

References

[1] R. Grycuk, P. Najgebauer, M. Kordos, M. M. Scherer, and A. Marchlewska, "Fast image index for database management engines," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 2, p. 113–123, 2020.

TABLE III
Experiment results for the proposed algorithm, performed on AIA images obtained from [18]. Due to lack of space, we present only a part of all queries. Dates range: 2012-01-01 : 2012-12-31.

Timestamp	RI	SI	PRI (TP)	FPRI (FP)	PNRI (FN)	Precision	Recall
2012-01-01 00:00:00	188	241	141	47	100	0.75	0.59
2012-01-04 15:00:00	358	481	320	38	161	0.89	0.67
2012-01-08 06:00:00	349	481	326	23	155	0.93	0.68
2012-01-16 05:00:00	386	481	343	43	138	0.89	0.71
2012-01-20 01:00:00	320	481	309	11	172	0.97	0.64
2012-01-24 22:00:00	352	481	310	42	171	0.88	0.64
2012-01-27 10:06:00	379	481	336	43	145	0.89	0.7
2012-02-01 16:12:00	336	481	324	12	157	0.96	0.67
2012-02-04 11:18:00	336	481	313	23	168	0.93	0.65
2012-02-06 02:18:00	376	481	321	55	160	0.85	0.67
2012-02-14 12:24:00	310	481	305	5	176	0.98	0.63
2012-02-21 17:24:00	334	481	329	5	152	0.99	0.68
2012-02-22 21:30:00	365	481	320	45	161	0.88	0.67
2012-02-26 20:30:00	365	481	325	40	156	0.89	0.68
2012-03-03 17:36:00	359	481	312	47	169	0.87	0.65
2012-03-09 16:36:00	365	481	328	37	153	0.9	0.68
2012-03-13 08:36:00	364	481	322	42	159	0.88	0.67
2012-03-18 16:36:00	343	481	318	25	163	0.93	0.66
2012-03-26 09:36:00	326	481	293	33	188	0.9	0.61
2012-03-30 06:36:00	337	481	319	18	162	0.95	0.66
2012-04-01 12:36:00	359	481	310	49	171	0.86	0.64
2012-04-04 12:42:00	365	481	322	43	159	0.88	0.67
2012-04-10 13:42:00	330	481	311	19	170	0.94	0.65
2012-04-13 22:48:00	349	481	315	34	166	0.9	0.65
2012-04-19 15:48:00	361	481	327	34	154	0.91	0.68
2012-04-24 16:48:00	380	481	331	49	150	0.87	0.69
2012-04-29 19:54:00	352	481	325	27	156	0.92	0.68
2012-05-06 13:00:00	331	481	316	15	165	0.95	0.66
2012-05-10 02:06:00	365	481	328	37	153	0.9	0.68
2012-05-14 10:12:00	359	481	329	30	152	0.92	0.68
2012-05-21 05:12:00	320	481	303	17	178	0.95	0.63
2012-05-29 00:12:00	380	481	345	35	136	0.91	0.72
2012-06-04 22:18:00	329	481	319	10	162	0.97	0.66
2012-06-06 05:18:00	375	481	328	47	153	0.87	0.68
2012-06-13 23:18:00	329	481	308	21	173	0.94	0.64
2012-06-18 00:24:00	344	481	329	15	152	0.96	0.68
2012-06-21 12:24:00	380	481	335	45	146	0.88	0.7
2012-07-07 22:30:00	372	481	327	45	154	0.88	0.68
2012-07-14 07:30:00	328	481	322	6	159	0.98	0.67
2012-07-30 20:42:00	317	481	302	15	179	0.95	0.63
2012-08-12 06:48:00	334	481	321	13	160	0.96	0.67
2012-08-22 20:54:00	342	481	332	10	149	0.97	0.69
2012-08-24 23:00:00	321	481	305	16	176	0.95	0.63
2012-09-01 09:12:00	376	481	331	45	150	0.88	0.69
2012-09-09 21:12:00	308	481	299	9	182	0.97	0.62
2012-09-14 09:18:00	339	481	326	13	155	0.96	0.68
2012-09-20 07:24:00	344	481	322	22	159	0.94	0.67
2012-09-25 08:30:00	333	481	312	21	169	0.94	0.65
2012-10-08 10:42:00	354	481	308	46	173	0.87	0.64
2012-10-17 22:48:00	336	481	313	23	168	0.93	0.65
2012-10-23 03:00:00	332	481	315	17	166	0.95	0.65
2012-10-29 19:06:00	362	481	320	42	161	0.88	0.67
2012-11-02 08:12:00	336	481	318	18	163	0.95	0.66
2012-11-11 05:12:00	379	481	338	41	143	0.89	0.7
2012-11-17 02:18:00	348	481	313	35	168	0.9	0.65
2012-11-21 00:24:00	376	481	338	38	143	0.9	0.7
2012-11-22 17:24:00	333	481	303	30	178	0.91	0.63
2012-11-25 12:24:00	346	481	300	46	181	0.87	0.62
2012-11-26 20:30:00	354	481	316	38	165	0.89	0.66
2012-12-08 22:42:00	368	481	345	23	136	0.94	0.72
2012-12-13 04:48:00	367	481	321	46	160	0.87	0.67
2012-12-26 14:00:00	355	481	310	45	171	0.87	0.64

TABLE IV
EXPERIMENT RESULTS FOR THE PROPOSED ALGORITHM, PERFORMED ON AIA IMAGES OBTAINED FROM [18]. DUE TO LACK OF SPACE, WE PRESENT ONLY A PART OF ALL QUERIES. DATES RANGE: 2017-01-01 : 2017-12-31.

Timestamp	RI	SI	PRI (TP)	FPRI (FP)	PNRI (FN)	Precision	Recall
2017-01-01 00:00:00	166	241	154	12	87	0.93	0.64
2017-01-08 11:00:00	348	481	317	31	164	0.91	0.66
2017-01-16 02:06:00	320	481	307	13	174	0.96	0.64
2017-01-20 12:12:00	360	481	327	33	154	0.91	0.68
2017-01-25 11:12:00	345	481	314	31	167	0.91	0.65
2017-01-27 03:18:00	377	481	334	43	147	0.89	0.69
2017-02-02 21:18:00	344	481	322	22	159	0.94	0.67
2017-02-08 14:18:00	342	481	318	24	163	0.93	0.66
2017-02-15 06:24:00	377	481	332	45	149	0.88	0.69
2017-02-24 02:24:00	361	481	321	40	160	0.89	0.67
2017-02-28 07:30:00	347	481	327	20	154	0.94	0.68
2017-03-03 04:36:00	351	481	324	27	157	0.92	0.67
2017-03-11 07:36:00	335	481	324	11	157	0.97	0.67
2017-03-16 15:42:00	373	481	326	47	155	0.87	0.68
2017-03-19 01:42:00	333	481	307	26	174	0.92	0.64
2017-03-22 22:48:00	340	481	318	22	163	0.94	0.66
2017-03-28 10:48:00	352	481	318	34	163	0.9	0.66
2017-04-05 05:54:00	335	481	326	9	155	0.97	0.68
2017-04-09 19:00:00	364	481	317	47	164	0.87	0.66
2017-04-21 04:12:00	378	481	332	46	149	0.88	0.69
2017-05-06 09:18:00	347	481	324	23	157	0.93	0.67
2017-05-11 17:18:00	343	481	316	27	165	0.92	0.66
2017-05-13 00:24:00	378	481	334	44	147	0.88	0.69
2017-05-26 18:30:00	344	481	337	7	144	0.98	0.7
2017-05-29 18:30:00	330	481	314	16	167	0.95	0.65
2017-06-04 21:36:00	341	481	323	18	158	0.95	0.67
2017-06-09 02:42:00	342	481	317	25	164	0.93	0.66
2017-06-20 20:54:00	356	481	316	40	165	0.89	0.66
2017-06-26 21:00:00	376	481	338	38	143	0.9	0.7
2017-07-16 15:00:00	321	481	310	11	171	0.97	0.64
2017-07-25 09:06:00	370	481	322	48	159	0.87	0.67
2017-08-05 21:12:00	370	481	330	40	151	0.89	0.69
2017-08-11 08:18:00	342	481	321	21	160	0.94	0.67
2017-08-13 18:24:00	348	481	333	15	148	0.96	0.69
2017-08-16 08:24:00	331	481	311	20	170	0.94	0.65
2017-08-24 13:30:00	333	481	316	17	165	0.95	0.66
2017-08-29 05:30:00	357	481	311	46	170	0.87	0.65
2017-08-30 23:36:00	357	481	319	38	162	0.89	0.66
2017-09-06 07:42:00	348	481	317	31	164	0.91	0.66
2017-09-09 22:48:00	358	481	343	15	138	0.96	0.71
2017-09-14 03:48:00	341	481	311	30	170	0.91	0.65
2017-09-15 17:54:00	347	481	299	48	182	0.86	0.62
2017-09-24 15:00:00	357	481	317	40	164	0.89	0.66
2017-10-02 16:06:00	358	481	324	34	157	0.91	0.67
2017-10-09 18:12:00	356	481	313	43	168	0.88	0.65
2017-10-14 00:18:00	359	481	332	27	149	0.92	0.69
2017-10-17 06:18:00	351	481	317	34	164	0.9	0.66
2017-10-19 09:24:00	324	481	314	10	167	0.97	0.65
2017-10-24 19:24:00	335	481	325	10	156	0.97	0.68
2017-10-31 11:24:00	328	481	317	11	164	0.97	0.66
2017-11-03 00:30:00	361	481	317	44	164	0.88	0.66
2017-11-10 10:30:00	363	481	319	44	162	0.88	0.66
2017-11-13 08:30:00	366	481	325	41	156	0.89	0.68
2017-11-15 04:36:00	368	481	321	47	160	0.87	0.67
2017-11-18 11:42:00	365	481	319	46	162	0.87	0.66
2017-11-21 13:48:00	330	481	325	5	156	0.98	0.68
2017-11-27 01:48:00	370	481	327	43	154	0.88	0.68
2017-12-02 17:54:00	358	481	320	38	161	0.89	0.67
2017-12-09 20:00:00	322	481	309	13	172	0.96	0.64
2017-12-15 08:00:00	349	481	330	19	151	0.95	0.69
2017-12-16 20:06:00	377	481	334	43	147	0.89	0.69
2017-12-22 12:12:00	368	481	322	46	159	0.88	0.67

[2] M. Korytkowski, R. Senkerik, M. M. Scherer, R. A. Angryk, M. Kordos, and A. Siwocha, “Efficient image retrieval by fuzzy rules from boosting and metaheuristic,” *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 1, p. 57–69, 2020.

[3] J. Banda, R. Angryk, and P. Martens, “Steps toward a large-scale solar image data analysis to differentiate solar phenomena,” *Solar Physics*, vol. 288, no. 1, pp. 435–462, 2013.

[4] J. M. Banda and R. A. Angryk, “Large-scale region-based multimedia retrieval for solar images,” in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2014, pp. 649–661.

[5] ——, “Scalable solar image retrieval with lucene,” in *2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 11–17.

[6] ——, “Regional content-based image retrieval for solar images: Traditional versus modern methods,” *Astronomy and computing*, vol. 13, pp. 108–116, 2015.

[7] D. Kempoton, M. Schuh, and R. Angryk, “Towards using sparse coding in appearance models for solar event tracking,” in *2016 19th International Conference on Information Fusion (FUSION)*, 2016, pp. 1252–1259.

[8] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 19–60, 2010.

[9] D. J. Kempton, M. A. Schuh, and R. A. Angryk, “Tracking solar phenomena from the sdo,” *The Astrophysical Journal*, vol. 869, no. 1, p. 54, 2018.

[10] ——, “Describing solar images with sparse coding for similarity search,” in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 3168–3176.

[11] E. Rafajłowicz, M. Wnuk, and W. Rafajłowicz, “Local detection of defects from image sequences,” *International Journal of Applied Mathematics and Computer Science*, vol. 18, no. 4, pp. 581–592, 2008.

[12] E. Rafajłowicz and W. Rafajłowicz, “Testing (non-) linearity of distributed-parameter systems from a video sequence,” *Asian Journal of Control*, vol. 12, no. 2, pp. 146–158, 2010.

[13] G. B. de Souza, D. F. da Silva Santos, R. G. Pires, A. N. Marananil, and J. P. Papa, “Deep features extraction for robust fingerprint spoofing attack detection,” *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 1, pp. 41–49, 2019.

[14] R. Salakhutdinov and G. Hinton, “Semantic hashing,” *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969 – 978, 2009, special Section on Graphical Models and Information Retrieval. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888613X08001813

[15] P. Duda, M. Jaworski, A. Cader, and L. Wang, “On training deep neural networks using a streaming approach,” *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 1, pp. 15–26, 2020.

[16] W. Wei, B. Zhou, D. Połap, and M. Woźniak, “A regional adaptive variational pde model for computed tomography image reconstruction,” *Pattern Recognition*, vol. 92, pp. 64–81, 2019.

[17] W. Wei, X.-L. Yang, B. Zhou, J. Feng, and P.-Y. Shen, “Combined energy minimization for image reconstruction from few views,” *Mathematical Problems in Engineering*, vol. 2012, 2012.

[18] A. Kucuk, J. M. Banda, and R. A. Angryk, “A large-scale solar dynamics observatory image dataset for computer vision applications,” *Scientific data*, vol. 4, p. 170096, 2017.

[19] E. R. Dougherty, “An introduction to morphological image processing,” *SPIE, 1992*, 1992.

[20] J. Serra, *Image analysis and mathematical morphology*. Academic Press, Inc., 1983.

[21] K. Kavitha and B. T. Rao, “Evaluation of distance measures for feature based image registration using alexnet,” *arXiv preprint arXiv:1907.12921*, 2019.

[22] M. Buckland and F. Gey, “The relationship between recall and precision,” *Journal of the American society for information science*, vol. 45, no. 1, p. 12, 1994.

[23] K. M. Ting, “Precision and recall,” in *Encyclopedia of machine learning*. Springer, 2011, pp. 781–781.

[24] S. Das, S. Garg, and G. Sahoo, “Comparison of content based image retrieval systems using wavelet and curvelet transform,” *The International Journal of Multimedia & Its Applications*, vol. 4, no. 4, p. 137, 2012.