*electronics*

MDPI

*Article*

# A Comparative Study of Reduction Methods Applied on a Convolutional Neural Network

**Aurélie Cools *, Mohammed Amin Belarbi and Sidi Ahmed Mahmoudi**

Faculty of Engineering, University of Mons, 7000 Mons, Belgium;
mohammedamin.belarbi@umons.ac.be (M.A.B.); sidi.mahmoudi@umons.ac.be (S.A.M.)
* Correspondence: aurelie.cools@umons.ac.be

**Abstract:** With the emergence of smartphones, video surveillance cameras, social networks, and multimedia engines, as well as the development of the internet and connected objects (the Internet of Things—IoT), the number of available images is increasing very quickly. This leads to the necessity of managing a huge amount of data using Big Data technologies. In this context, several sectors, such as security and medicine, need to extract image features (index) in order to quickly and efficiently find these data with high precision. To reach this first goal, two main approaches exist in the literature. The first one uses classical methods based on the extraction of visual features, such as color, texture, and shape for indexation. The accuracy of these methods was acceptable until the early 2010s. The second approach is based on convolutional neuronal networks (CNN), which offer better precision due to the largeness of the descriptors, but they can cause an increase in research time and storage space. To decrease the research time, one needs to reduce the size of these vectors (descriptors) by using dimensionality reduction methods. In this paper, we propose an approach that allows the problem of the "curse of dimensionality" to be solved thanks to an efficient combination of convolutional neural networks and dimensionality reduction methods. Our contribution consists of defining the best combination approach between the CNN layers and the regional maximum activation of convolutions (RMAC) method and its variants. With our combined approach, we propose providing reduced descriptors that will accelerate the research time and reduce the storage space while maintaining precision. We conclude by proposing the best position of an RMAC layer with an increase in accuracy ranging from 4.03% to 27.34%, a decrease in research time ranging from 89.66% to 98.14% in the function of CNN architecture, and a reduction in the size of the descriptor vector by 97.96% on the GHIM-10K benchmark database.

**Keywords:** CBIR; image indexation; features extraction; dimensionality reduction; CNN; deep learning; RMAC; RMAC+; MS-RMAC

**Citation:** Cools, A.; Belarbi, M.A.; Mahmoudi, S.A. A Comparative Study of Reduction Methods Applied on a Convolutional Neural Network. *Electronics* **2022**, *11*, 1422. https://doi.org/10.3390/electronics11091422

Academic Editors: Zheng Wang, Jian Zhao, Hong Liu, Zhun Zhong

Received: 13 April 2022
Accepted: 27 April 2022
Published: 28 April 2022



## 1. Introduction

Nowadays, several devices can collect data due to people using their phones to produce images and videos every day before sharing them with friends and family on different social network platforms, such as Instagram and TikTok, and on a wide variety of subjects. These images are of increasingly high quality.

Other images can be present in different domains (medicine, spatial) which need to be found quickly and efficiently [1]. This is becoming an important area of research, which is called image retrieval [2]. This field of research is useful for different sectors such as video surveillance [3], for example, to identify cars where a faster search is desired with high accuracy, identification of people [3,4]; e-commerce, to identify shopping recommendations based on photos; various applications to recognize flowers, mushrooms, etc. To solve this problem, research engines appeared a few years ago. The first one was based on keywords, but this was not enough to index effectively. Later, a content-based research

engine appeared, called CBIR, which stands for "content-based image retrieval". This is a way of querying a database within a query image; the goal is to find similar images. The user aims to upload a query image into the research engine. The latter extracts the properties of the query image into a descriptor vector and compares them with the image properties in the database. Finally, the research engine returns similar images to the user by ranking them from the most similar to the least similar.

In the literature, there are two main methods: hand-made and deep learning methods. The hand-made methods are based on image descriptors, such as color histogram, grey level covariance matrix, local binary pattern, or scale-invariant feature transform. Deep learning methods are well known in the field, especially the convolutional neural network (CNN), and are widely used to address this problem thanks to their precision potential, but they are computationally intensive since the image descriptors are large, which results in significant computation time and storage issues. To reduce image descriptors, we propose a comparative study of dimensionality reduction methods with a focus on those appropriate to CNN descriptors.

This paper is organized as follows:

- Related work:

In this section, we will study CBIR and consider the different existing methods to extract visual features classically and within deep learning. Then, we will check the different possibilities to reduce the size of the descriptors depending on the extraction methods. At the end, we will expose the methods adapted to the dimensionality reduction in the extracted descriptors within CNNs by analyzing them and, more particularly, the regional maximum activation of convolutions method (RMAC) and two of its variants.

- Proposed approach:

In this section, we explore and implement the different RMAC methods to compare and analyze in detail the process and impact of each method on CBIR algorithms. We detail the results obtained on different architectures for the GHIM-10k database according to different criteria and discuss the results obtained for these different methods by explaining these results according to the architectures. As a result, we propose the best appropriate position for the dimension reduction layer within a convolutional neural network.

- Discussion and conclusion:

We will draw conclusions based on our results for the different methods and highlight the advantages of these methods.

- Perspectives and future work:

We conclude this paper by putting forward different perspectives of this work and future potential works.

## 2. Related Work

For indexing and searching images, content-based image retrieval search engines (CBIR) are the most widely used since they are able to extract visual information from the image without the need for keywords. Indeed, manual annotations with keywords can lead to malfunctions. For example, some similar words may have different meanings. Conversely, different words can be used to express the same thing. Content-based engines avoid these problems but have other disadvantages such as vector size and increased search time.

### *2.1. CBIR System*

CBIR is a system that allows the retrieval of similar images based on a query image, where a descriptor vector is extracted based on the query image features and compared to the feature vectors of the images in the database. In order to extract the features of an image, there are different methods—the classical ones and those based on deep learning.

The classical methods are based on feature extraction methods and the comparison of descriptors is then done via similarity measures [5]. To define images, a large part of CBIRs focus on primitive [6], also called low-level, features such as color, shape, texture [6–9], and spatial features [8,9]. These features are extracted with methods such as color histograms [6,9,10], the grey level co-occurrence matrix (GLCM) [5,7], or the Local Binary Pattern (LBP) [7,11], to name only the best known. In addition to color, shape, and texture features, there are also points of interest in an image, which are information-rich points such as a corner or a boundary. These points are robust to illumination variations, rotation, and scale changes. The most well-known descriptors for highlighting points of interest in the literature are: Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) [4].

For several years, neural networks have shown their efficiency for image feature extraction and, more particularly, for convolutional neural networks (CNN), which lead to high precision thanks to the convolutional layers [12]. Moreover, some works use the combination of features or neural network architectures to increase the accuracy of CBIR [4,13–15]. This leads to a high increase in the descriptors' size and research time.

Given the amount of data to be managed, the usefulness of CBIR systems is no longer in doubt in many sectors. However, with the increase in image quality such as 4k and 8K resolutions, the size of the descriptor vectors is big, which can negatively influence the research time and the storage space. This is accentuated by the increase in the size of the databases, which are more crowded due to the ability of connected objects to capture more and more images and the ability to store them easily.

We therefore face a trade-off between precision and reasonable research time. To deal with this problem, several dimensionality reduction methods have emerged.

Another type of architecture, called transformers, start to be used for image recognition with comparable performance to CNNs. Notice that transformers were mainly used for natural language processing [16] and are slowly being applied to computer vision [17]. According to the authors in [18], CNNs and transformers are complementary to reach better performance. In [19], the authors mention that transformers can also be competitive to provide small descriptor vectors. In this paper, we focus our work on dimensionality reduction applied to CNN only.

### *2.2. Dimensionality Reduction*

In [20], the authors show that the choice of the dimensionality reduction technique depends on several factors, including the type of data. It should reduce the dimensionality without the loss of important information, and it should gain in precision and computation time.

For classical indexing methods, we can mention the Fisher Vector and the Vector of Locally Aggregated Descriptors (VLAD). These vectors allow different features to be assembled into a compact vector [13]. Principal component analysis (PCA) also allows the dimension of the feature vector to be reduced [2,7,21]. The authors in [22] studied the different reductions related to PCA applied to scale-invariant feature transform (SIFT) and Speeded Up Robust Features (SURF) descriptors on Wang (http://wang.ist.psu.edu/docs/related/, accessed on 20 March 2021) and Coil100 (https://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php, accessed on 20 March 2021) databases and concluded that the optimal reduction was at 70%. These authors conclude that PCA can effectively reduce the computational cost and maintain high accuracy [22]. The above methods are not suitable for deep neural networks.

#### 2.2.1. Extraction of a Feature Vector from a CNN

The features of an image can be extracted from different locations in the network. Two methods regularly appear: either from the last convolutional layer, as illustrated in Figure 1, or from fully connected layers. According to the authors in [4], the deeper the network, the more powerful learning capability it can provide, thus extracting high-level

abstract and semantic data. Extracting features from fully connected layers can have limitations as explained by the authors in [4]. Some authors take the fully connected layers as a feature vector and compare the similarities with the Euclidean distance or the cosine distance [12,23].

In [24], the authors use the penultimate fully connected layer of the AlexNet as a feature extractor. In [22,25], the authors extract features from the last three convolutional layers. It should be noted that the latter approach has limitations due to the type of fully connected layer. Indeed, in a fully connected layer, each neuron is linked to all neurons of the previous layer and fully connected layers have a global field. This leads to two limitations: firstly, a lack of spatial information, and secondly, a lack of local geometric invariance, i.e., it affects the robustness of image transformations such as truncation and occlusion [4].

In [4], the authors derive the feature vector of an image via convolutional layers. This is often the last layer that retains more structural detail. Usually, the robustness of these features improves after pooling, as these layers preserve more local information such as corners or edges. A neuron in a convolutional layer is linked to a local region of the input image. This smaller receptive field ensures that features are more robust against occlusion and truncation [26].

It has been shown that the pooling strategy on the last convolutional layer performs better than on hidden layers and fully connected layers [4] because these layers reduce the sensitivity to distortions such as scale changes [26].

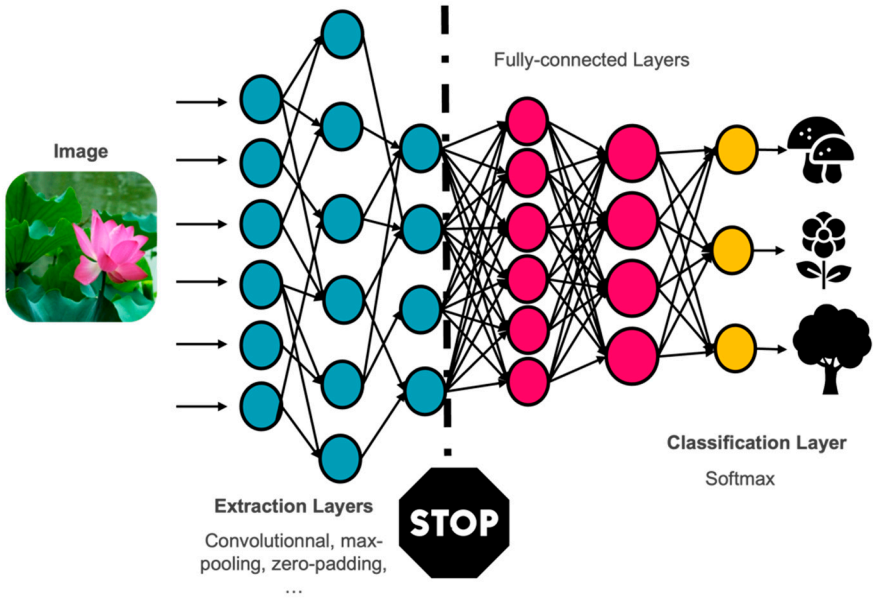


**Figure 1.** Illustration of position for extraction feature vectors after the last convolutional layer.

A specific method proposed by Tolias et al. in [27], called RMAC, reduces the size of the descriptors extracted from a CNN.

#### 2.2.2. RMAC

Another method to reduce the dimension of the feature vector seems to be differentiated in the field of content-based image indexing and retrieval with CNN. It was presented by Tolias et al. in [27], called the RMAC descriptor, for the Regional Maximal Activation of convolution.

According to the authors in [27], RMACs are state-of-the-art descriptors for image retrieval. The authors explain that it is a representation that encodes and aggregates multiple regions of the image into a compact and dense representation. An image runs

through a pre-trained CNN on ImageNet (https://image-net.org/update-mar-11-2021.php, accessed on 28 April 2022), the output of the last convolutional layer, then goes through a max-pooling over different regions of different scales to obtain a feature vector for each region. These vectors are then L2-normalized, reduced within PCA, normalized again with L2, and finally aggregated by summing the whole, normalized to obtain the final vector [28]. The authors in [29] indicate that this is the most used descriptor in image retrieval problems.

This method relies on a maximum convolution activation layer called the Maximum Activations of Convolutions (MAC) layer which is also used by the authors in [30], who add such a layer following the convolutional layers that max-pool over each subregion $(2 \times 2)$ in order to more efficiently represent the image features.

They consider that a pre-trained CNN is a fully convolutional network, i.e., all fully connected layers are discarded [27]. They consider square regions, R, at different sizes, L, on the image, I. At the largest scale, the region size is equal to the minimum between the width and height of the image, I. The regions are chosen uniformly so that the overlap between two consecutive regions is as close to 40% as possible [27].

At each different scale, some regions are uniformly sampled, which results in 20 regions when L = 3 [27]. Figure 2 shows us these regions, which are represented by the dots as their centers.

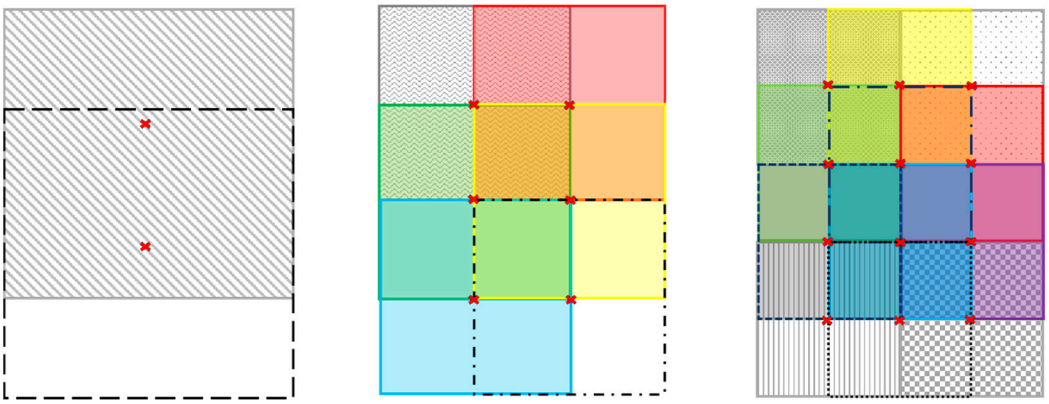

**Figure 2.** Illustration of RMAC regions.

Figure 3 shows these regions as an example. There are two regions in the first representation, six in the second, and twelve in the last. It is the sum of these regions that constitutes the twenty regions of the method.

A descriptor vector is created for each region, on which an L2-normalization is applied, followed by a PCA, and again an L2-normalization [12]. Finally, the RMAC vector is obtained by the sum all the features of the regions into a single vector (Equation (1) [12]:

$$F_j = \sum_{i=1}^{N} f_{R_i} = \left[ \sum_{i=1}^{N} f_{R_i,1} \cdots \sum_{i=1}^{N} f_{R_i,k} \cdots \sum_{i=1}^{N} f_{R_i,K} \right]^T \quad (1)$$

where $j$ is the layer of convolutional maps and $N$ is the total number of regions. The final dimension of the RMAC descriptor is equal to the number of feature channels, $K$. The authors in [27] conclude that they have better results than the MAC method with the same vector dimension and without additional computational costs.

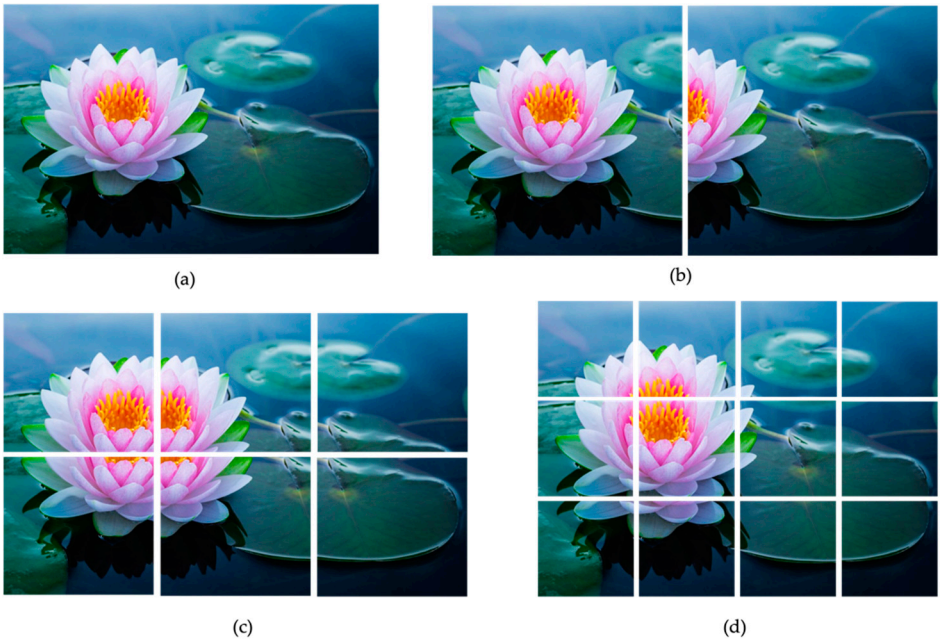


**Figure 3.** Example of RMAC regions. (**a**) Query image, (**b**) Two first areas of RMAC method, (**c**) Six next areas of RMAC method, (**d**) Twelve last areas of RMAC method.

#### 2.2.3. RMAC+

In [31], the authors propose a variant of the RMAC method called RMAC+ and apply it for landscape recognition. There are two main differences between RMAC+ and RMAC, in the division of the zones and in the search method.

- Area allocation

They first propose building the multiresolution descriptors on the resized images, increasing the dimensions of the feature maps in order to have more features and more local maxima compared to the classical method [31]. This method is connected to a new region detector that detects fifteen regions (not necessarily square), unlike RMAC which has twenty squared regions.

The first level covers the whole image (yellow in Figure 4). On the second level, it extracts two regions that are the largest possible in relation to the image, and are either in relation to the x-axis or in relation to the y-axis, according to the longest one (blue in Figure 4). The image is then cut into six regions (green in Figure 4), and finally, six other regions with respect to the axis opposite to the previous step (red in Figure 4).

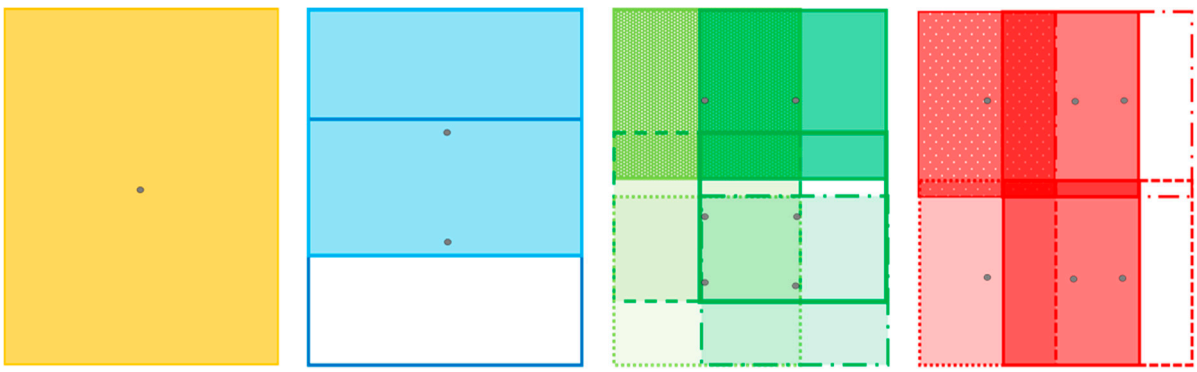

**Figure 4.** Illustration region's method RMAC+.

It is better to overlay the regions rather than not covering the whole image otherwise we might lose essential information [31]. You can find an example of RMAC+ regions in Figure 5.

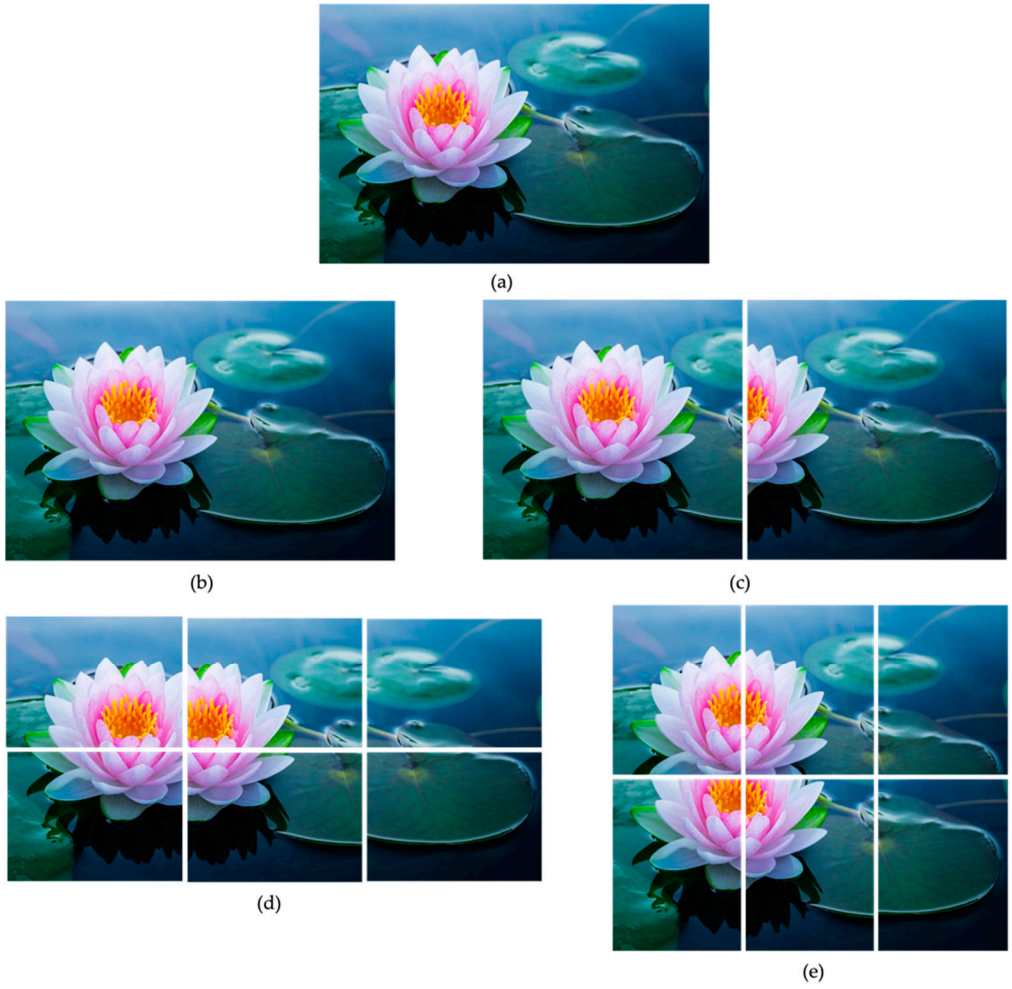


**Figure 5.** Example of region's method RMAC+. (**a**) Query image, (**b**) First area of RMAC+, (**c**) Two next areas of RMAC+ method, (**d**) Six next areas from RMAC+ method and (**e**) Six last areas of RMAC + method

- Search method

The authors in [31] propose an improvement of the search method by comparing the vector of each region with the vector of the query image. According to [31], this new method takes more time but can improve the performance of CBIR.

#### 2.2.4. MS-RMAC

In [32], the authors propose another variant of RMAC. It is a method proposing a multi-scale approach called MS-RMAC which consists of a multitude of RMAC descriptors built on different convolution layers as illustrated in Figure 6. According to the authors in [32], their MS-RMAC method gives better results than the RMAC proposed in [27]. The authors in [33] propose another variant of MS-RMAC with the InceptionV3 architecture and a post-processing of the vector descriptor before calculating the ranking loss.

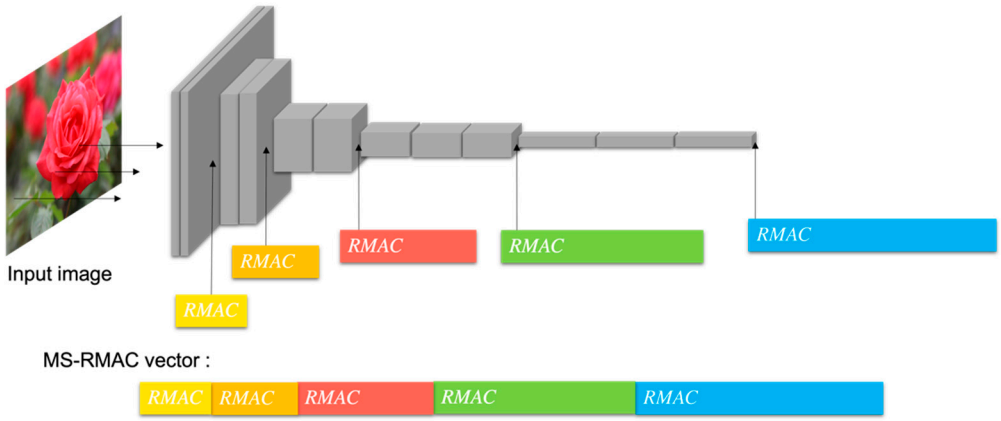


**Figure 6.** Illustration of MS-RMAC method on VGG 16 architecture.

### *2.3. Performance Measures*

The performance of a CBIR can be measured within precision [2,6] and recall [7], and according to different authors, these are the most widely used methods [8,11,34]. A high precision value implies that the system returns more relevant images than irrelevant ones, while a high recall value means that the system returns most of the relevant images [7]. In general, the precision value will decrease when the recall value increases. If both values remain high, the CBIR can be considered ideal.

On the other hand, the authors in [6,34] use the evaluation measure called the mean average precision (MaP), which is an average of the average precision (aP) that is one of the most widely used methods [35] to evaluate the quality of returned images. The average precision takes into consideration the recall and precision.

## 3. Proposed Approach

The methods presented above are mainly applied to the VGG16 network. There is no specific information for other types of architecture. We complement this method with our approach, which proposes a position for different architectures.

The proposed approach consists of several steps: transfer learning, feature extraction, the application of reduction methods, and analysis. Experimentations were conducted using the cluster of the Faculty of Engineering at UMONS with the following configuration:

- 48 GB of RAM;
- Processor: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz with 32 cores;
- Graphics card: GTX 1080ti with 12GB of RAM.

Training the CNNs represented the most consuming step in terms of computation time, which is about one day per CNN. Notice that the application of dimensionality reduction caused a slight increase (between 15 and 20%) in computation time when applied after the last convolutional layer only.

At the software level:

- Python 3.5;
- Tensorflow 2.x;
- Keras.

### *3.1. Dataset*

We used a common image dataset to demonstrate the effect of our approach on the problem of large-scale image retrieval. We used the GHIM-10k (http://www.ci.gxnu.edu.cn/cbir/Dataset.aspx) database, which contains 10,000 images presented within 20 classes, as illustrated in Figure 7.

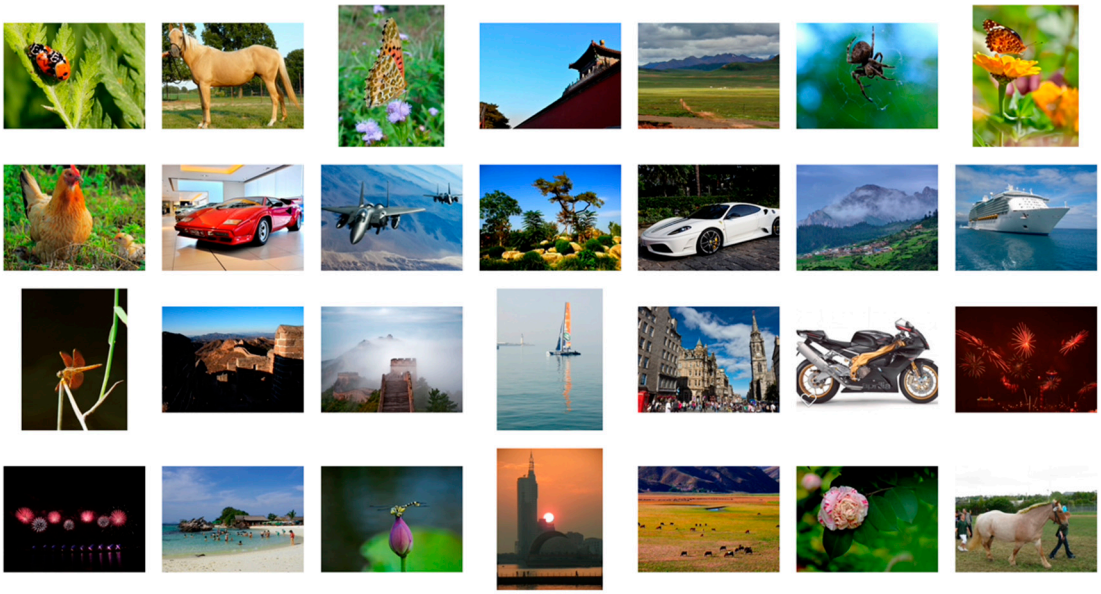

**Figure 7.** Illustration of the GHIM-10k dataset.

### *3.2. Transfer Learning*

Transfer learning is applied to different architectures for classification applied to the GHIM-10K database: Xception, VGG16, VGG19, ResNet50, InceptionV3, InceptionResNetV2, MobileNet, DenseNet121, DenseNet169 and DenseNet201. Five architectures will be retained for this research by choosing the architectures that perform well in terms of accuracy and by varying the type of architecture. The results of this transfer learning are presented in Table 1 with a MaP based on the TOP100, with nine query images contained in three different classes.

**Table 1.** Results of classification in precision.

| Models | Accuracy (%) | Loss | VAL_ACC (%) | VAL_LOSS |
|---|---|---|---|---|
| Xception | 99.13 | 0.038 | 99.00 | 0.036 |
| VGG16 | 99.31 | 0.022 | 99.80 | 0.008 |
| VGG19 | 98.94 | 0.031 | 99.33 | 0.021 |
| ResNet50 | 99.43 | 0.023 | 99.87 | 0.004 |
| MobileNet | 98.50 | 0.055 | 100 | 0.003 |
| InceptionV3 | 99.27 | 0.029 | 99.93 | 0.002 |
| InceptionResNetV2 | 98.91 | 0.045 | 99.80 | 0.014 |
| DenseNet121 | 99.11 | 0.035 | 100 | 0.002 |
| DenseNet169 | 99.76 | 0.013 | 100 | 0 |
| DenseNet201 | 99.64 | 0.019 | 100 | 0.001 |

On the basis of the results illustrated in Table 2, the models retained are VGG16, MobileNet, Xception, DenseNet169, and ResNet50. We have categorized these architectures into "simple architectures" which include straight architectures such as VGG16 and MobileNet, i.e., the connections between the layers follow one after the other, and "complex architectures" where the input of a layer comes from different layers, as is the case for DenseNet169, ResNet50, and Xception. Based on these results, we can conclude that the DenseNet architecture is the most suitable for GHIM database search.

Another essential criterion in the CBIR is the search time which depends, in particular, on the size of the feature vector. The search time and the size of the vector can be found in Table 2. Based on this criterion, the architecture VGG16 is by far the best.

**Table 2.** Comparison of models based on time search (in seconds) and vector size.

| | Models | Search | Vector Size |
|---|---|---|---|
| Simple architecture | VGG16 | 0.58 | 25,088 |
| | MobileNet | 1.04 | 50,176 |
| Complex architecture | ResNet50 | 2.09 | 100,352 |
| | Xception | 4.29 | 100,352 |
| | DenseNet169 | 1.67 | 81,536 |

### *3.3. Position of the Initial Feature Vector Extraction*

The convolutional neural networks (CNN) are composed of two parts. One part concerns the feature extraction, while the second concerns the classification composed of the fully connected layers and the classifier layer. The remainder of this paper only focuses on the first part concerning feature extraction. We removed the classification layer and the fully connected layers before applying the dimension reduction, as shown in Figure 1.

### *3.4. Reduction Method Application*

#### 3.4.1. RMAC Method

According to [28], the RMAC layer must be applied after the last convolutional layer of the network. This is illustrated in Figure 8.

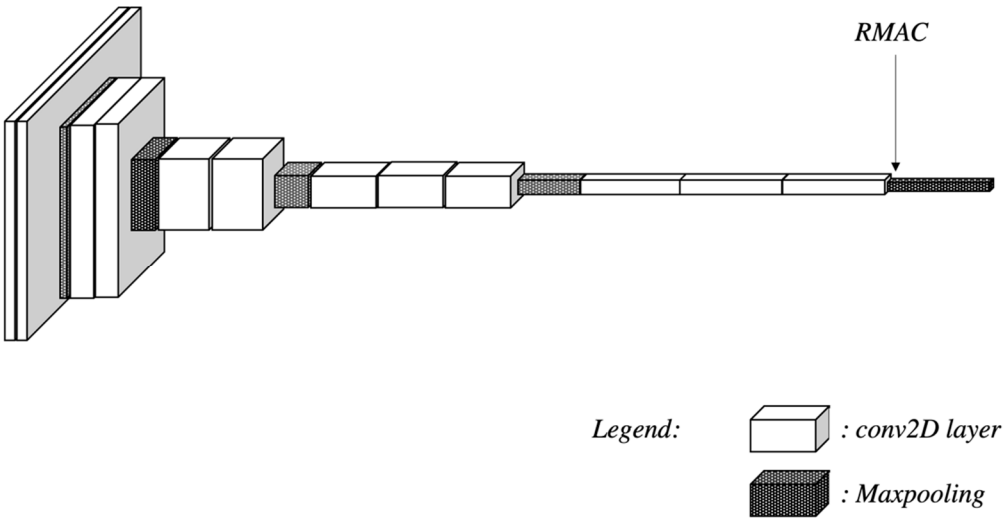


**Figure 8.** Position of the RMAC layer in VGG16.

For the five selected models, this corresponds to positioning the RMAC layer after the layers mentioned in Table 3.

**Table 3.** Previous layer of the RMAC position based on models.

| Models | Layer Before RMAC Position |
|---|---|
| Xception | conv2d_3 |
| VGG16 | block5_conv3 |
| ResNet50 | conv5_block3_3_conv |
| MobileNet | conv_pw_13 |
| DenseNet169 | conv5_block32_2_conv |

The results of the addition of the RMAC layer after the last convolutional layer can be found in Table 4. Two architectures, namely VGG16 and MobileNet, which are "simple

architectures", keep good performances in terms of MaP while the "complex architectures" give poor results.

**Table 4.** MaP with RMAC layer after the last convolutional layer (in %).

| | **Simple Architectures** | | |
|---|---|---|---|
| | **VGG16** | **MobileNet** | |
| MaP (in %) | 99.64 | 97.75 | |
| | COMPLEX ARCHITECTURES | | |
| | ResNet50 | Xception | DenseNet169 |
| MaP (in %) | 79.13 | 4.20 | 0 |

Analysis of RMAC Results

In order to improve the results obtained in Table 4, we analyzed the impact of the position of the layer RMAC on the "complex architectures".

By analyzing the position of the RMAC layer in simple and complex architectures, we assumed that, for this to work well, the RMAC layer should not be placed in a block, but instead after one. In other words, the results of a layer pass entirely to the following layer until the end of the network, while the other architectures have parallel blocks. There are two other possible positions for Xception and one other possible position for ResNet50 and DenseNet169.

Alternative Position in ResNet50

The architecture ResNet50 is composed of a repetition of blocks constituted by several convolutions in series, as well as in parallel, as illustrated by Figure 9.

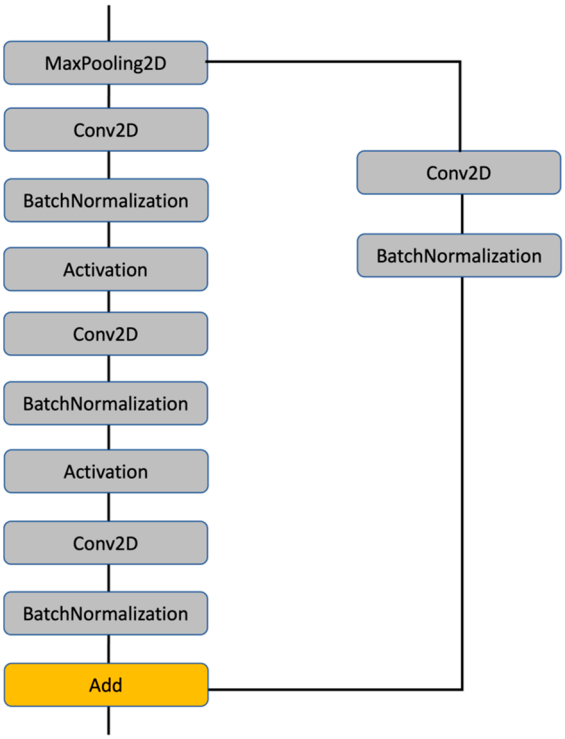


**Figure 9.** Partial illustration of ResNet50 architecture.

By adding the RMAC layer to the last convolution layer (conv2D), as the first method proposed, this layer is added to one of these blocks and lost a part of the information. According to our hypothesis, the RMAC layer must be placed at the end of the block, joining the two sides of the one in question after the additional layer.

For the ResNet50 network, the new location considered for the RMAC layer is the conv5_block3_add layer which is at the end of the last block of the network. The new position of the RMAC layer improves the results, which can be seen in Table 5.

**Table 5.** MaP with RMAC layer after new position (%) in complex architectures.

| Architectures | Layers | MaP |
|---|---|---|
| Xception | Add_11 | 0 |
| | Block_14 | 90.32 |
| ResNet50 | Add | 85.89 |
| DenseNet169 | Concatenate | 99.94 |

Alternative Position in DenseNet169

DenseNet169 connects several layers to a single concatenation layer, as shown in Figure 10. We then tried the same approach as ResNet50, i.e., by placing the RMAC layer at the end of the block at the concatenated layer. This also gave better results, which are shown in Table 5.

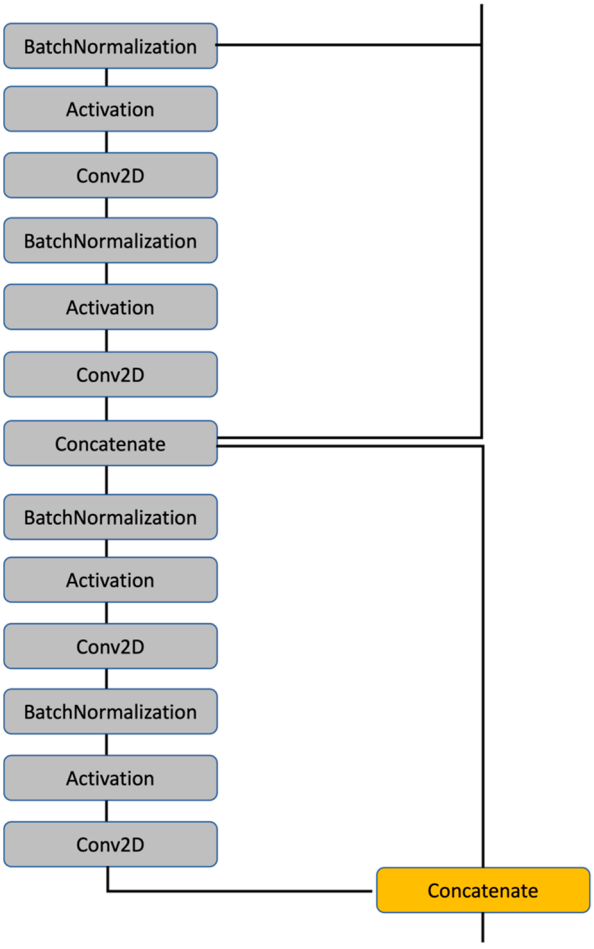


**Figure 10.** Partial illustration of the DenseNet169 architecture.

Alternative Position in Xception

In the structure of the blocks, the Xception architecture strongly resembles the ResNet50architecture. An alternative position for the RMAC layer is at the end of the last block of the network, like for ResNet 50, which corresponds to the add_11 layer, which corresponds to the end of the block where the last convolutional layer is located. As can

be seen in Table 5, which contains the results of the changes in the location of the RMAC layer, placing the RMAC layer after add_11 does not improve the result for Xception.

There are the main differences between the Xception and ResNet50 architectures: in ResNet50, after the additional layer, there is no more "convolution" layer, which is contrary to Xception, which contains, after this additional layer, two separable convolution layers, as illustrated in Figure 11. Based on these analyses, the RMAC layer is placed after the last separable convolution layer, block14_sepconv2, which is outside the blocks of the network. This gave better results by reaching a MaP of 90.32%, as shown in Table 5.

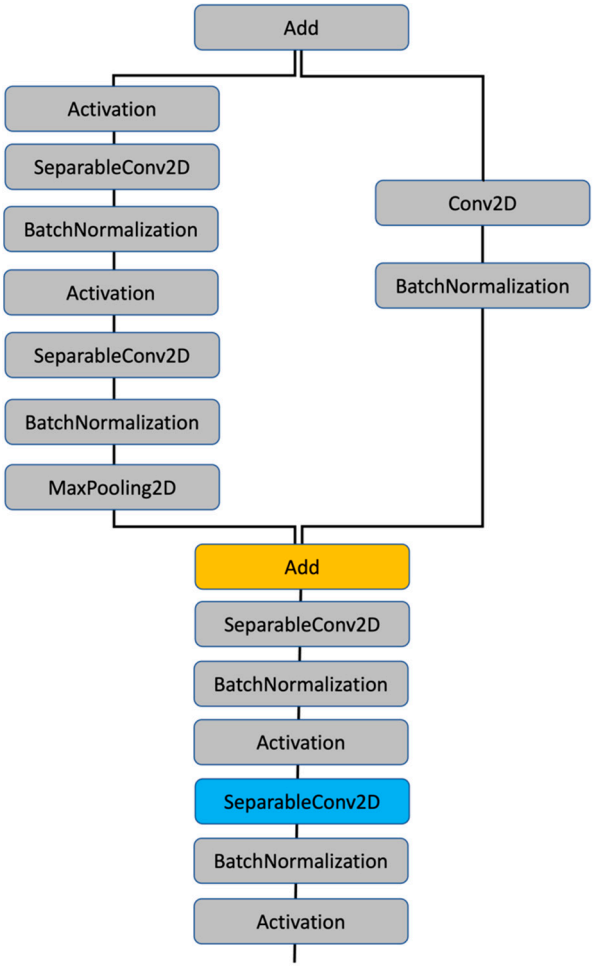


**Figure 11.** Partial illustration of Xception architecture.

Performance of the Proposed Approach for the RMAC Method

The synthesis of the results obtained with the RMAC method can be found in Table 6. According to this table, we can see that the best performances are obtained based on MaP by DenseNet169, and based on the search time by VGG16. The differences in time can be explained by the size of the different vectors. The size of the RMAC descriptor depends on the depth of the layer on which it can be applied.

**Table 6.** Synthesis of RMAC methods results.

| | | Simple Architectures | | Complex Architectures | | |
|---|---|---|---|---|---|---|
| | **MODELS** | **VGG16** | **MobileNet** | **ResNet50** | **Xception** | **Dense-Net169** |
| CLASSIC | MaP | 95.78 | 77.32 | 67.45 | 78.27 | 99.86 |
| | Search Time | 0.58 | 1.04 | 2.08 | 4.29 | 1.67 |
| | Size | 25,088 | 50,176 | 100,352 | 100,352 | 81,536 |
| RMAC * | MaP | 99.64 | 97.75 | 85.89 | 90.32 | 99.94 |
| | Search Time | 0.06 | 0.07 | 0.08 | 0.08 | 0.07 |
| | Size | 512 | 1024 | 2048 | 2048 | 1664 |
| RMAC+ | MaP | 92.72 | 59.93 | 39.31 | 28.85 | 51.00 |
| | Search Time | 7.41 | 7.91 | 9.52 | 9.57 | 9.10 |
| | Size | 512 | 1024 | 2048 | 2048 | 1664 |
| MS-RMAC * | MaP | 99.64 | 98.09 | 89.31 | 91.40 | 99.94 |
| | Search Time | 0.06 | 0.08 | 0.14 | 0.10 | 0.14 |
| | Size | 1472 | 3584 | 8192 | 3584 | 8000 |

* Those results are which with best layer found.

Comparing the search times between the classical version and the RMAC method, we conclude that the dimensionality reduction RMAC method speeds up the search for all models. Nevertheless, we argue that the acceleration is stronger for "complex architectures". We can therefore deduce that the RMAC method is suitable for all models, but offers even better performance on networks giving large feature vectors.

Compared to the dimension reduction, the results obtained confirm the efficiency of the RMAC method. The RMAC method can therefore improve the MAP and reduce the search time.

3.4.2. Analysis of RMAC+ Results

As a reminder, the RMAC+ method extracts features at the end of the network, before the fully connected layers, as illustrated in Figure 12.

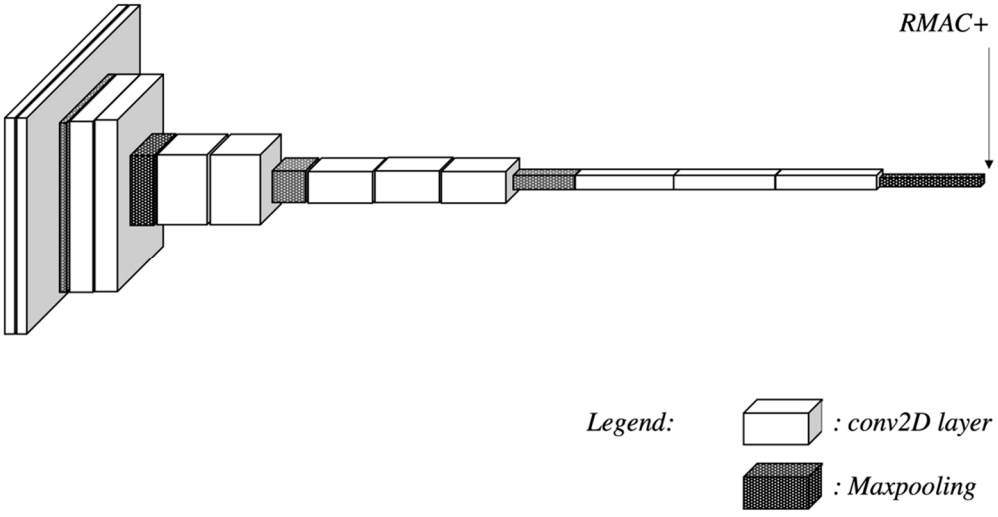


**Figure 12.** Illustration of RMAC+ position on VGG16 architecture.

The synthesis of the results obtained with the RMAC+ method in terms of the MaP for each architecture is presented in Table 6. In view of these results, only VGG16 maintains a MaP higher than 90%, while the other models see their performance deteriorate.

Plotting these results as a graph of average precision versus vector size, we conclude that the RMAC+ method is more appropriate for small vectors, as shown in Figure 13.

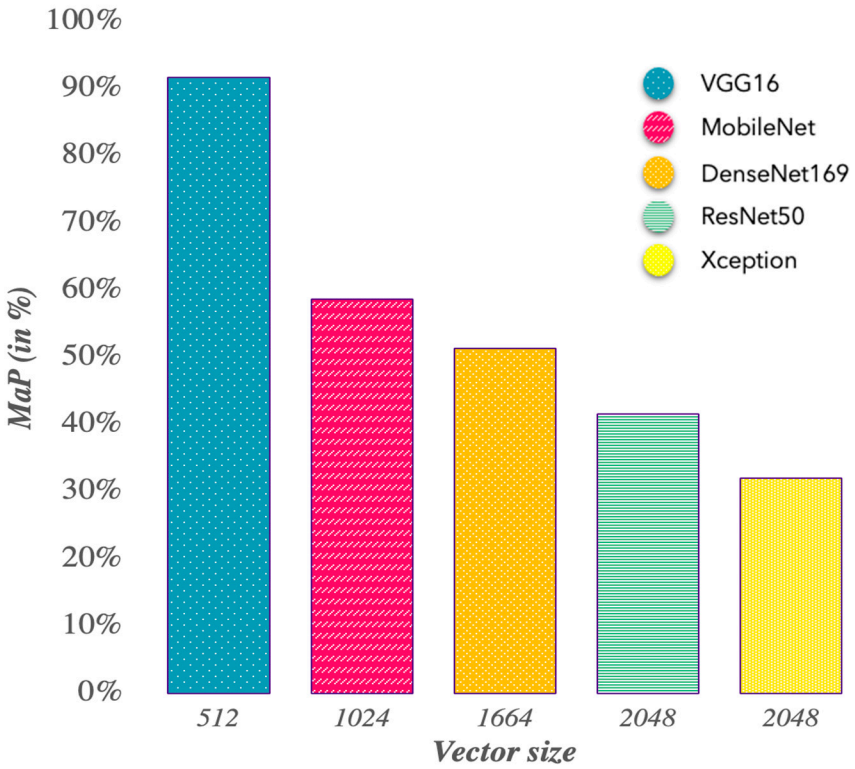


**Figure 13.** Results MaP in the function of RMAC+ vector size.

This can be explained by the fact that when the feature vectors are larger, as the regions are fixed, more features are merged into one region. We lose a lot of information about large vectors. In order to confirm this hypothesis, we trained the MobileNet model with a layer, reducing the size of the output vector to 512 instead of 1024. The MaP then goes from 59.93% to 62.97%.

In the RMAC+ algorithm, there is an operation to compute the square root of the image feature matrix in which there are negative values. It was therefore impossible to calculate this root. We therefore opted for a solution with complex numbers in order to keep the complete information. This has a negative impact on the search time compared to the classical search. The authors of the article [31] already pointed out this increase in the search time compared to the RMAC method and their research focused on images concerning landscapes which are very different from the images used in our case.

### 3.4.3. Analysis of MS-RMAC Results

As a reminder, this method involves inserting several RMAC layers at different places in the network, as illustrated in Figure 14. To do this, we rely on the best layers found using our approach for the RMAC method.

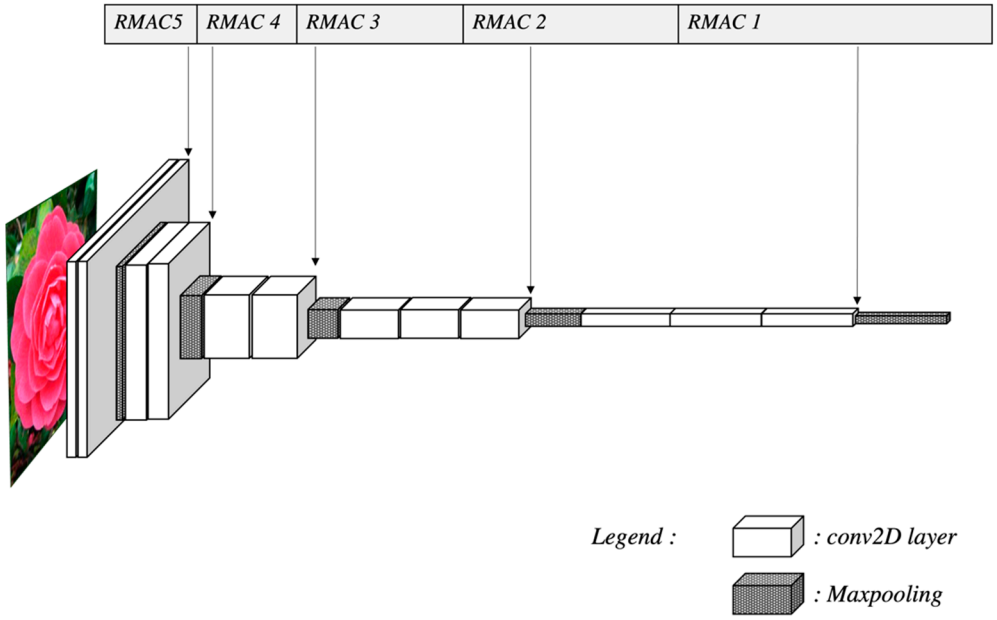


**Figure 14.** Illustration of localization RMAC layers to form MS-RMAC in VGG16.

For the VGG16 network, the different RMAC layers were inserted at the last convolutional layer of each block, as the author did in [32]. For the VGG16network, which contains five blocks, five RMAC layers were applied. We can deduce that the RMAC layers are to be applied at each end of the block for the MS-RMAC method.

Nevertheless, for some networks, such as DenseNet169, this would correspond to adding 83 RMAC layers and would give a feature vector size of 10,560, which requires a lot of time for the indexing. We therefore decided to limit the number of added layers to a maximum of five, which already corresponds to a feature vector size of 8000 for the DenseNet169 network, which corresponds to 75.75% of the size of the vector initially thought. For the ResNet50 and DenseNet169 networks, we added the five RMAC layers at the end of the last five blocks of the network. For the MobileNet architecture, we applied the same principle as for VGG16, i.e., we applied the RMAC layers to the last convolution layers of the last blocks, limiting their number to five.

On the other hand, for the Xception network, our research led us to add the RMAC layer to the separable convolution layer. There are only two at the end of the network. We will therefore apply only RMAC to these two layers to form the MS-RMAC vector. The corresponding results can be found in Table 6.

We have highlighted that "simple architectures", such as VGG16 and MobileNet, lose slightly in MaP, while "complex architectures", such as Xception, ResNet50, and DenseNet169, see their MaP increase with our proposition. Regarding the time, it remains stable compared to RMAC for simple architectures, while it increases for complex architectures. It increases to a lesser extent for Xception because we have added only two layers compared to the other models where we have added five layers. For these complex architectures, the search time doubled. MS-RMAC does not achieve a higher score in terms of MAP on straight architectures compared to RMAC. On the other hand, it increases the quality of the search compared to the RMAC method for complex architectures.

To complete our research, we also tried to combine architectures and descriptors to improve accuracy. In view of the previous results, we applied only the RMAC method to these combinations. For the combination of descriptors (of features), we applied the RMAC layer in each CNN independently and then concatenated the descriptor vector. For the combination of architectures, we merged the architectures through an added layer and applied RMAC after this layer. Our experiments showed that merging the features provided better performance in terms of accuracy than merging the architectures.

Nevertheless, the results of this combination were, in terms of accuracy, comparable to the RMAC method on a unique CNN but increased the search time significantly.

## *4.* Discussion

Table 7 and Figure 15 take up the improvements of mean average precision (MaP), the reduction in time, and in vector size compared to the classical method. These variations are calculated according to Equation (2):

$$Variation\ (\%) = \frac{Method\ to\ evaluate\ -\ Method\ classic}{Method\ classic}.100 \quad (2)$$

**Table 7.** Comparison of improvements based on the increase in MaP, reduction in search time, and vector size (in%).

| | RMAC | | | RMAC+ | | | MS-RMAC | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Improv. Map** | **Red. Time** | **Red. Size** | **ImproV. Map** | **Red. Time** | **Red. Size** | **Improv. Map** | **Red. Time** | **Red. Size** |
| SIMPLE ARCHITECTURES | | | | | | | | | |
| VGG16 | + 4.03 % | − 89.66 % | − 97.96 % | - | - | − 97.96 % | + 3.40 % | − 89.66 % | − 94.13 % |
| MobileNet | +26.42 % | − 93.27 % | − 97.96 % | - | - | − 97.96 % | + 24.21 % | − 92.31 % | − 92.86 % |
| COMPLEX ARCHITECTURES | | | | | | | | | |
| ResNet50 | + 27.34 % | − 96.17 % | − 97.96 % | - | - | − 97.96 % | + 28.54 % | − 93.30 % | − 91.84 % |
| Xception | +15.40 % | − 98.14 % | − 97.96 % | - | - | − 97.96 % | + 18.69 % | − 98.14 % | − 96.43 % |
| DenseNet169 | + 0.08 % | − 95.81 % | − 97.96 % | - | - | − 97.96 % | + 0.10 % | − 91.62 % | − 90.19 % |

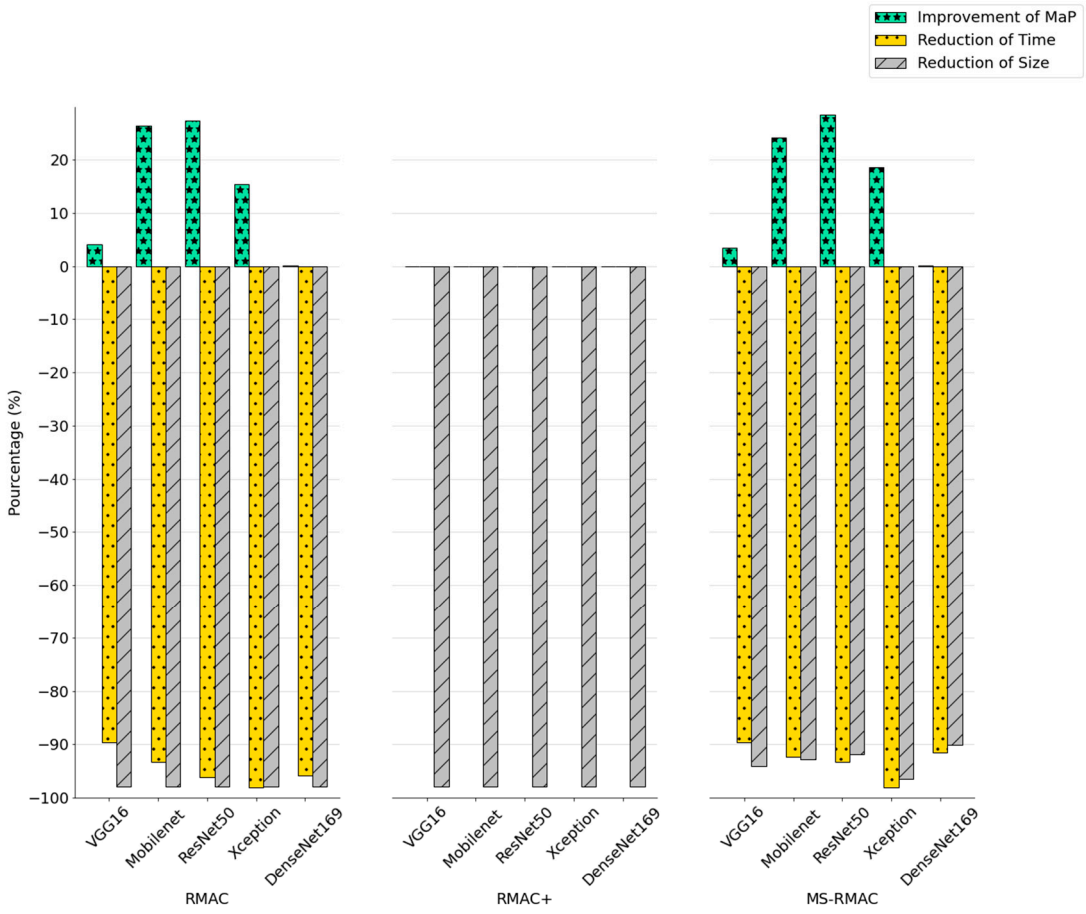


**Figure 15.** Comparison of methods improvements based on increasing MaP, reduction in search time, and reduction in size vector (%).

The size of the vectors of the RMAC and RMAC+ methods is identical. Nevertheless, the performances are not in time or in MaP. The differences between RMAC and RMAC+ are at two levels. Firstly, the location of feature extraction is not identical, as illustrated in Figures 8 and 12, and RMAC works on 20 square regions (as illustrated in Figure 3), while RMAC+ works on 15 square or rectangular regions (as illustrated in Figure 5). Moreover, the other difference is in the search mode: in RMAC+, the comparison is done on the basis of the feature vectors per region and not on the descriptor vector of the whole image. It leads 15 regions to be compared three times, i.e., 45 regions for an image in the database.

The results of MS-RMAC are comparable to those of RMAC. In terms of precision, the MS-RMAC method on simple architectures does not bring any added value according to our experiments applied to the GHIM-10k dataset, which is contrary to complex architectures, for which MS-RMAC has a slight positive impact on the MaP. Regarding the size of the vectors, those of MS-RMAC are more important than RMAC since it is the fusion of several RMAC layers at different places of the network, but this method still leads to the size of the classical search vector being reduced by more than 90%. The size of the vector is defined thanks to the sum of the sizes of the RMAC vectors. In view of the size of the vector MS-RMAC, the search time increased, as the authors in [32] had announced, particularly for complex architectures, such as DenseNet169 and ResNet50, which saw their search time almost double compared to the RMAC method. Nevertheless, MS-RMAC allows the search time to be reduced by a minimum of 90% compared to the classical search. We can conclude that some methods are more adapted to certain architectures and according to the user's needs.

## 5. Conclusions

To conclude this paper, we can therefore put forward the following elements for each method. RMAC extract features based on twenty regions. For the latter, we proposed an improvement in the location of the RMAC layer.

The most suitable position of an RMAC layer for simple architectures is after the last convolutional layer. For complex architectures, however, there are two possibilities: either there is no more convolution (of any kind) after the blocks, in which case the most suitable position is after the last block, Or, if there are convolutions after the last block, then the most suitable position is after the last convolution. This improvement led to an increase in the mean average precision (MaP) ranging from 4% to 27%, except for DenseNet169. Regarding time, the gains have a reduction in time ranging from 89.66% to 98%. The vector size reduction has a reduction of 97%.

The RMAC+ approach uses 15 regions to extract features from three images of different scales. Our approach led to the size being reduced by the 97% vector, concluding that this method maintained a better precision on small vectors: the smaller the vector initially, the more the precision will be maintained.

MS-RMAC is an extension of RMAC that applies several layers of RMAC at different locations of the network. Using the more suitable locations found in our analyses of the RMAC method and limiting the number of layers added to five maximum resulted in an improvement for the MAP, ranging from 3% to 28% except for DenseNet169. For the reduction in search time and vector size, the results are a reduction in time ranging from 92% to 98% and a decrease in vector size ranging from 90% to 96%.

## 6. Perspectives and Future Work

For future work, we plan to exploit new architectures within different positions of the RMAC layer. The transformers will be also studied as alternatives or complement to CNN architectures with a customized method of dimensionality reduction. We share our code on: https://github.com/ACOOLS/Reduction-dimension-RMAC.git. (accessed on 28 April 2022)

**Author Contributions:** Conceptualization, A.C.; methodology, A.C.; formal analysis, A.C.; investigation, A.C.; writing—original draft preparation, A.C.; writing—review and editing, M.A.B. and S.A.M.; visualization, A.C.; supervision, M.A.B. and S.A.M.; funding acquisition, S.A.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** GHIM-10k: http://www.ci.gxnu.edu.cn/cbir/Dataset.aspx (accessed on 20 March 2021).

**Acknowledgments:** Conceptualization: A.C.; Formal analysis, A.C.; Investigation, A.C.; Methodology, A.C.; Supervision, M.A.B. and S.A.M.; Visualization, A.C.; Writing – original draft, A.C.; Writing – review & editing, M.A.B. and S.A.M. The authors would like to express their gratitude to academic editors and publishers for the editing of this paper thanks to the waiver offer.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| CNN | Convolutional neural network |
| CBIR | Content-based image retrieval search engines |
| RMAC | Regional maximum activation of convolutions |
| PCA | Principal component analysis |
| SIFT | Scale-invariant feature transform |
| SURF | Speeded up robust features |
| VLAD | Vector of Locally Aggregated Descriptors |
| MS-RMAC | Multi-scale regional maximum activation of convolutions |
| MaP | Mean average precision |
| aP | Average precision |

## References

1. Amato, G.; Carrara, F.; Falchi, F.; Gennaro, C.; Vadicamo, L. Large-scale instance-level image retrieval. *Inf. Process. Manag.* **2020**, *57*, 102100.
2. Hussain, D.M.; Surendran, D. The efficient fast-response content-based image retrieval using spark and MapReduce model framework. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *12*, 4049–4056.
3. Zhao, G.; Zhang, M.; Liu, J.; Li, Y.; Rong Wen, J. AP-GAN: Adversarial patch attack on content-based image retrieval systems. *GeoInformatica* **2020**, *26*, 347–377.
4. Chen, W.; Liu, Y.; Wang, W.; Bakker, E.; Georgiou, T.; Fieguth, P.; Liu, L.; Lew, M.S. Deep Image Retrieval: A Survey. *arXiv* **2022**, arXiv:2101.112822021.
5. Yu, J.; Xie, G.; Li, M.; Xie, H.; Yu, L. Beauty Product Retrieval Based on Regional Maximum Activation of Convolutions with Generalized Attention. In Proceedings of the 27th ACM International Conference on Multimedia, Lisboa, Portugal, 21–25 October 2019; pp. 2553–2557. https://doi.org/10.1145/3343031.3356065.
6. Sadeghi-Tehran, P.; Angelov, P.; Virlet, N.; Hawkesford, M.J. Scalable database indexing and fast image retrieval based on deep461learning and hierarchically nested structure applied to remote sensing and plant biology. *J. Imaging* **2019**, *5*, 33.
7. Rana, S.P.; Dey, M.; Siarry, P. Boosting content based image retrieval performance through integration of parametric & nonparametric approaches. *J. Vis. Commun. Image Represent.* **2019**, *58*, 205–219.
8. Haji, M.S.; Alkawaz, M.H.; Rehman, A.; Saba, T. Content-based image retrieval: a deep look at features prospectus. *Int. J. Comput. Vis. Robot.* **2019**, *9*, 14–38

9. Zheng, Q.; Tian, X.; Yang, M.; Wang, H. Differential Learning: A Powerful Tool for Interactive Content-Based Image Retrieval. *Eng. Lett.* **2019**, *27*, EL_27_1_23.
10. Boucher, A.; Le, T.L. Comment extraire la sémantique d'une image? In Proceedings of the Conference Internationale Sciences Electroniques, Technologies de l'Information et des Telecommunications (SETIT'05), Sousse, Tunisia, 27–31 March 2005; pp. 295–306.
11. Sotoodeh, M.; Moosavi, M.R.; Boostani, R. A novel adaptive LBP-based descriptor for color image retrieval. *Expert Syst. Appl.* **2019**, 127, 342–352.
12. Qi, L.; Lu, X.; Li, X. Exploiting spatial relation for fine-grained image classification. *Pattern Recognit.* **2019**, 91, 47–55.
13. Alzu'bi, A.; Amira, A.; Ramzan, N. Learning transfer using deep convolutional features for remote sensing image retrieval. *Int. J. Comput. Sci.* 2019, 46, 1–8.
14. Hameed, I.M.; Abdulhussain, S.H.; Mahmmod, B.M.; Hussain, A. Content Based Image Retrieval Based on Feature Fusion and Support Vector Machine. In Proceedings of the 2021 14th International Conference on Developments in eSystems Engineering (DeSE), Sharjah, United Arab Emirates, 7–10 December 2021; pp. 552–558. https://doi.org/10.1109/DeSE54285.2021.9719539.
15. Wang, W.; Jiao, P.; Liu, H.; Ma, X.; Shang, Z. Two-stage content based image retrieval using sparse representation and feature fusion. *Multimedia Tools and Applications,* **2022**, https://doi.org/10.1007/s11042-022-12348-7.
16. Khan, S.H.; Naseer, M.; Hayat, M.; Zamir, S.W.; Khan, F.S.; Shah, M. Transformers in Vision: A Survey. *arXiv* **2021**, arXiv:2101.01169v5.
17. Li, C.; Yang, J.; Zhang, P.; Gao, M.; Xiao, B.; Dai, X.; Yuan, L.; Gao, J. Efficient Self-supervised Vision Transformers for Representation Learning. *arXiv* **2021**, arXiv:2106.09785.
18. Park, N.; Kim, S. How Do Vision Transformers Work? *arXiv* **2022**, arXiv:2202.06709. https://doi.org/10.48550/ARXIV.2202.06709.
19. El-Nouby, A.; Neverova, N.; Laptev, I.; Jégou, H. Training Vision Transformers for Image Retrieval. *arXiv* **2021**, arXiv:2102.05644.
20. Wei, S.; Liao, L.; Li, J.; Zheng, Q.; Yang, F.; Zhao, Y. Saliency inside: learning attentive CNNs for content-based image retrieval. *IEEE Trans. Image Process.* **2019**, *28*, 4580–4593.
21. Khan, U.A.; Din, S.M.U.; Lashari, S.A.; Saare, M.A.; Ilyas, M. Cowbree: A novel dataset for fine-grained visual categorization. *Bull. Electr. Eng. Inform.* **2020**, *9*, 1882–1889.
22. Qian, Q.; Jin, R.; Zhu, S.; Lin, Y. Fine-grained visual categorization via multi-stage metric learning. In Proceedings of the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3716–3724.
23. Jain, S.; Dhar, J. Image based search engine using deep learning. In Proceedings of the 2017 Tenth International Conference on Contemporary Computing (IC3), Noida, India, 10–12 August 2017; pp. 1–7.
24. Shah, A.; Naseem, R.; Iqbal, S.; Shah, M.A.; et al. Improving cbir accuracy using convolutional neural network for feature extraction. In Proceedings of the 2017 13th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 27–28 December 2017; pp. 1–5.
25. Ramanjaneyulu, K.; Swamy, K.V.; Rao, C.S. Novel CBIR System using CNN Architecture. In Proceedings of the 2018 3rd International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 15–16 November 2018; pp. 379–383.
26. Wang, L.; Wang, X. Model and metric choice of image retrieval system based on deep learning. In Proceedings of the 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Datong, China, 15–17 October 2016; pp. 491390–491395.
27. Tolias, G.; Sicre, R.; Jégou, H. Particular object retrieval with integral max-pooling of CNN activations. *arXiv* **2015**, arXiv:1511.058792015.
28. Sun, M.; Yuan, Y.; Zhou, F.; Ding, E. Multi-attention multi-class constraint for fine-grained image recognition. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 805–821.
29. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
30. Feng, Y.; Lan, L.; Zhang, X.; Xu, C.; Wang, Z.; Luo, Z. AttResNet: Attention-based ResNet for Image Captioning. In Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence, Sanya, China, 21–23 December 2018; pp. 1–6.
31. Magliani, F.; Prati, A. An accurate retrieval through R-MAC+ descriptors for landmark recognition. In Proceedings of the 12th International Conference on Distributed Smart Cameras, Eindhoven, The Netherlands, 3–4 September 2018; pp. 1–6. https://doi.org/10.1145/3243394.3243686.
32. Li, Y.; Xu, Y.; Wang, J.; Miao, Z.; Zhang, Y. MS-RMAC: Multiscale Regional Maximum Activation of Convolutions for Image Retrieval. *IEEE Signal Process. Lett.* **2017**, *24*, 609-613. https://doi.org/10.1109/LSP.2017.2665522.
33. Alappat, A.L.; Nakhate, P.; Suman, S.; Chandurkar, A.; Pimpalkhute, V.; Jain, T. CBIR using Pre-Trained Neural Networks. *arXiv* **2021**, arXiv:2110.14455.
34. Valem, L.P.; Pedronette, D.C.G. Unsupervised selective rank fusion for image retrieval tasks. *Neurocomputing* **2020**,*377*, 182–199.
35. Kanwal, K.; Ahmad, K.T.; Khan, R.; Abbasi, A.T.; Li, J. Deep Learning Using Symmetry, FAST Scores, Shape-Based Filtering and Spatial Mapping Integrated with CNN for Large Scale Image Retrieval. *Symmetry* **2020**, *12*, 612.