applied sciences

MDPI

Article

Development and Experimental Evaluation of Machine-Learning Techniques for an Intelligent Hairy Scalp Detection System

Wei-Chien Wang [1], Liang-Bi Chen [2,*] and Wan-Jung Chang [2,*]

[1] Science and Engineering Faculty, Queensland University of Technology, Brisbane 4000, Australia; cchain.5@gmail.com

[2] Department of Electronic Engineering, Southern Taiwan University of Science and Technology, Tainan 71005, Taiwan

* Correspondence: liangbi.chen@gmail.com (L.-B.C.); allenchang@stust.edu.tw (W.-J.C.)

Received: 14 May 2018; Accepted: 21 May 2018; Published: 23 May 2018

Featured Application: Deep learning, decision tree, linear discriminant analysis (LDA), support vector machines (SVMs), k-nearest neighbors algorithm (K-NN), and ensemble learning are evaluated for detecting hairy scalp problems. To the best of our knowledge, we are the first case study to apply modern machine learning to the diagnosis and analysis of hairy scalp issues.

Abstract: Deep learning has become the most popular research subject in the fields of artificial intelligence (AI) and machine learning. In October 2013, *MIT Technology Review* commented that deep learning was a breakthrough technology. Deep learning has made progress in voice and image recognition, image classification, and natural language processing. Prior to deep learning, decision tree, linear discriminant analysis (LDA), support vector machines (SVM), k-nearest neighbors algorithm (K-NN), and ensemble learning were popular in solving classification problems. In this paper, we applied the previously mentioned and deep learning techniques to hairy scalp images. Hairy scalp problems are usually diagnosed by non-professionals in hair salons, and people with such problems may be advised by these non-professionals. Additionally, several common scalp problems are similar; therefore, non-experts may provide incorrect diagnoses. Hence, scalp problems have worsened. In this work, we implemented and compared the deep-learning method, the ImageNet-VGG-f model Bag of Words (BOW), with machine-learning classifiers, and histogram of oriented gradients (HOG)/pyramid histogram of oriented gradients (PHOG) with machine-learning classifiers. The tools from the classification learner apps were used for hairy scalp image classification. The results indicated that deep learning can achieve an accuracy of 89.77% when the learning rate is 1×10^{-4}, and this accuracy is far higher than those achieved by BOW with SVM (80.50%) and PHOG with SVM (53.0%).

Keywords: deep learning; machine learning; support vector machine (SVM); images classification; images recognition; hairy scalp diagnosis and analysis

1. Introduction

In recent years, machine-learning techniques have been widely used in computer vision, image recognition, stock market analysis, medical diagnosis, natural language processing, voice/speech recognition, etc. Machine learning is an aspect of artificial intelligence (AI) that represents another widely used term for AI. Therefore, AI research has shifted from reasoning as the most vital aspect to knowledge and then learning as the most important aspects, which represents a natural and distinct

sequence of events. Machine learning has become a methodology of realizing AI, and it also represents a method of resolving issues associated with AI. Historically, huge quantities of data had to be analyzed to accomplish many different tasks, and such analyses represented a methodology used for policy making and modeling; however, machine learning provides a more effective and productive replacement methodology for acquiring knowledge.

Machine learning can progressively improve forecast modeling functions and capabilities and utilize relevant data to assist with policy formulation. Therefore, this approach has become an increasingly important tool in computer science-related research and has played a progressively vital role in people's daily lives. Machine learning has provided more advanced forms of junk email filters, better software for voice/speech recognition, and reliable internet search engines, among other benefits. Deep learning has attracted increasing attention in academia and industry and represents the most popular research direction in the fields of AI and machine learning. Even *MIT Technology Review* commented in October 2013 that deep learning was a breakthrough technology [1]. Deep learning has made progress in voice and image recognition, image classification, and natural language processing. Prior to deep learning, support vector machines (SVM) represented a popular approach to solving classification problems.

In this paper, we use the diagnosis and analysis of hairy scalps as a case study for machine-learning techniques. In this case study, we implemented, evaluated, and compared the deep learning method with the ImageNet-VGG-f model Bag of Words (BOW), with machine learning classifiers, and histogram of oriented gradients (HOG)/pyramid histogram of oriented gradients (PHOG) with machine-learning classifiers. We selected scalp state detection as our case study because people have many hairy scalp lesions caused by work pressure, long working hours, and a lack of scalp care, among other reasons.

In the hair salon industry, dermatology clinics and medical clinics, human methods of hairy scalp state detection are frequently employed. The professional education and training costs in these industries are very high. Moreover, the accuracy of hairy scalp state recognition varies among individuals and does not follow a set of criteria.

In this paper, we test whether machine learning technology can be applied to hairy scalp detection. A machine learning-based hairy scalp detector can automatically recognize the state of the hairy scalp. Moreover, machine learning can continue to train learning algorithms to increase the accuracy of scalp detection. We believe that machine learning-based AI image processing methods should be able to effectively solve the aforementioned hairy scalp detection problem. By installing machine-learning technology into a scalp state detector, the use of human-based assessments and the resulting errors can be reduced. To the best of our knowledge, this is the first study to apply modern machine-learning techniques to the diagnosis and analysis of hairy scalp problems.

The remainder of this paper is organized as follows. Section 2 introduces the preliminary assessment of machine-learning techniques. Section 3 contains a review of previous works on the design and development of image classification and recognition using machine-learning techniques. Section 4 presents the machine-learning techniques evaluated for diagnosing and analyzing hairy scalps. Section 5 describes the experimental results and provides a related discussion. Section 6 presents our conclusions and plans for future work.

2. Preliminaries

Figure 1 describes the principles [2] of building a machine-learning system. Machine-learning technology as a predictive modeling system can be divided into four parts: preprocessing, learning, evaluation, and prediction. Generally, the raw data and format of images cannot be directly used for calculations because such data could contain a considerable amount of useless information that may have a negative impact on the performance of the learning algorithms. Hence, the preprocessing of raw data plays an import role within every application of machine learning systems. In this work, the raw data are scalp images, and we try to capture the key features.

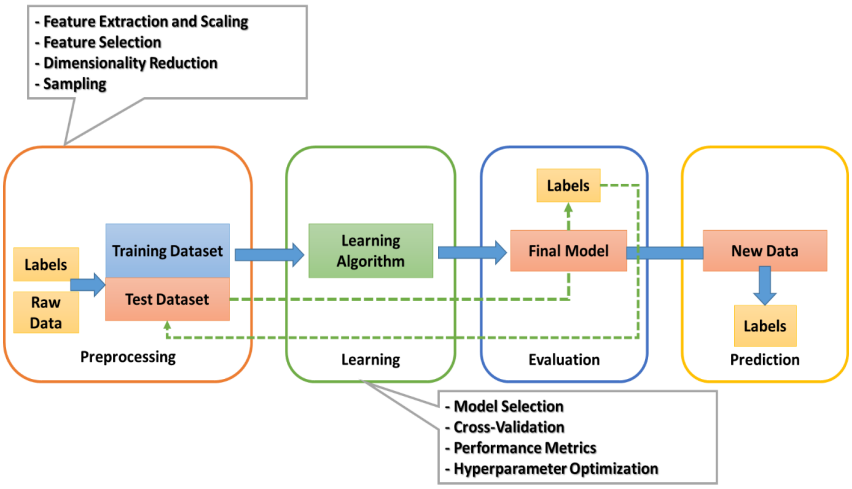


Figure 1. Typical machine-learning system [2].

The key features are usually the color, luminance, and density of hair. Features are selected to meet particular restrictions so that they perform well when using machine-learning algorithms. A good machine-learning algorithm predicts good results in both training and new datasets. The datasets are usually automatically divided into two parts (training and testing datasets). The training dataset is for training and optimizing the models, and the testing dataset is for evaluating the efficiency of the model.

Each machine-learning algorithm has its own advantages for solving specific problems, and one learning algorithm cannot manage all problems. Hence, to train and discover the best models, different learning algorithms could be applied to the same dataset. Comparisons of the training data results for each learning algorithm can be used to identify the most suitable learning model.

In the 1980s, multilayer perceptron (MLP) was a very popular machine learning technique, especially for speech recognition and image recognition. However, since the 1990s, MLP has encountered strong competition from the simpler support vector machines (SVM). Recently, due to the success of deep learning, MLP has regained attention. As we know, modern deep-learning techniques were based on the MLP structure.

MLP includes at least one hidden layer (in addition to one input layer and one output layer). Compared to the single-layer perceptron (SLP), which can only learn linear functions, MLP can learn non-linear functions. Figure 2 shows a simple structure of an MLP with a hidden layer. Please note that all connections are weighted, but only three weights (w0, w1, and w2) are marked in Figure 2. A simple structure of an MLP is introduced as follows.

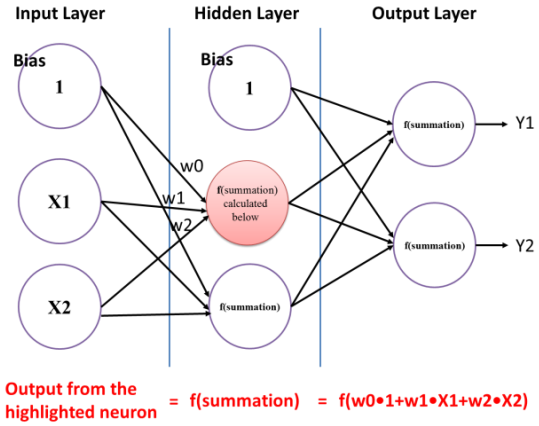


Figure 2. A simple structure of a multilayer perceptron (MLP) with a hidden layer.

Input layer: The input layer has three nodes. The offset node value is 1. The other two nodes take external inputs from X1 and X2 (all are digital values based on the input data set). As discussed above, no calculation is performed on the input layer; thus, the output of the input layer node is 1, and X1 and X2 are passed to the hidden layer.

Hidden layer: The hidden layer also has three nodes. The offset node output is 1. The output of the other two nodes of the hidden layer depends on the output of the input layer (1, X1, and X2) and the weight attached to the connection (boundary). Figure 2 shows the calculation of an output in a hidden layer (highlighted). The output calculations of the other hidden nodes are the same. Please note that "f" refers to the activation function. These outputs are passed to the nodes of the output layer.

Output layer: The output layer has two nodes, receives input from the hidden layer, and performs calculations similar to the highlighted hidden layer. These calculated values (Y1 and Y2), which are the result of the calculation, are the outputs of the MLP.

As a result, given a set of features X = (x1, x2, . . .) and a goal Y, an MLP can learn the relationship between features and goals for the purpose of classification or regression.

To assess the effectiveness of a given model, the accuracy of different models is compared. In this work, we have applied several popular machine-learning algorithms to analyze and diagnose hairy scalp images. First, we adopt a manual method of classifying hairy scalps and comparing the classification of the model accuracy to determine the quality of recognition.

Consequently, test datasets can be used to test the proposed models. Then, suitable models can be selected based on the training dataset. The performance of the test dataset compared with that of an untested dataset must be determined along with the error rate and performance. Subsequently, optimal models can be applied to predict new and future data.

3. Related Works

Machine learning-based techniques are widely discussed, studied and applied for image classification, image recognition, and object detection in many fields [3–21]. The related application cases of machine learning-based image detection and classification are introduced as follows.

For traffic applications [3–8,19], Lousier and Abdelkrim [3] proposed a bag of features (Bove)-based machine learning framework for image classification, and this assessed the performance of training models using different image classification algorithms on the Caltech 101 images [4]. These authors also adopted the proposed BoF-based machine-learning framework to identify stop sign images for applying the trained classifier in a robotic system. Ahmed et al. [5] presented text recognition that adopted convolutional neural networks (CNNs) as their deep-learning classifier for detecting and recognizing Arabic text. The error rate of their proposed recognition methodology was 15% using cursive script scene data.

Jagannathan et al. [6] implemented an embedded system-based object detection and classification mechanism that adopted a commercial SoC solution. The main components of this commercial SoC solution are fixed/floating-point dual DSP cores, a fully programmable VisionAccelerationPac (EVE), dual ARM-based Cortex M4 cores, and an image signal processor. In this work, the authors combined two methodologies, an AdaBoost cascade classifier with 10 HOG features for object detection and a 7-layer CNN classifier for objects classification. The implemented methodologies can achieve accuracies of 74.6% for pedestrian object detection, 79.4% for vehicle object detection, 79.6% for traffic sign object detection, and 89.6% for traffic sign objects classification.

Du et al. [7] proposed an end-to-end deep learning model that adopted the Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) architecture to predict real-time vehicular ego-motion for an autonomous driving system. The error rate of their proposed CNN-LSTM-based model for multiple ego-motion classification was 0.0417. Pop et al. [8] proposed different cross-modality CNN-based deep-learning training approaches for pedestrian recognition, and they consisted of a correlated model, an incremental model, and a particular cross-modality model.

For hyperspectral image classification applications [9–14], Ermushev and Balashov [9] developed a complex machine-learning technique for target detection from ground radar images called CTDCM (complex ground radar target detection and classification method), which was based on a three-step analysis that included a learning procedure, data pre-processing examination, and target-labeling step coupled with classification. The classification error rate of their proposed CTDCM was 12%.

Zhao and Du [10] proposed a spectral-spatial feature-based classification framework that integrated deep-learning techniques with dimension reductions for hyperspectral image classification. This work overcomes the insufficient discovery of objects that present considerable variations of shape in fixed detection windows. Zhong et al. [11] designed a supervised deep-learning framework that alleviated the declining accuracy of deep-learning models. The proposed work classified many agricultural and urban hyperspectral imagery data sets (Indian Pines, Kennedy Space Center, and University of Pavia).

Qader et al. [12] classified the types of vegetation extracted by satellite-based phonological characteristics in Iraq and achieved an overall accuracy of 85%. Chen et al. [13] applied exclusive and hierarchical relationships to enhance the classification accuracy of multiple-label scenes. In this work, the authors combined these two relationships with a LSTM to form an accurate CNN-based scene classifier.

For the other applications [15–21], Zhang et al. [15] proposed a covariance descriptor that combined visual and geometric information. Moreover, this work integrated a classification framework with dictionary learning for the object recognition of 3D point clouds. Remez et al. [16] presented a full CNN-based deep learning architecture for image de-noising that uses the splitting scheme to achieve sub-optimization. Nasr et al. [21] used multi-class SVMs that classified facial images for robotic applications. This work adopted BoF as a face representation. In their work, scale-invariant feature transform (SIFT) features were replaced by speeded up robust features (SURF) for rapid and accurate extractions. Moreover, SURF was also used for selecting interesting points.

Besides, a few hairy scalp issues were also discussed and researched [22–24], Shih and Lin [22] developed hair segmentation and counting algorithms which were based on an unsupervised mechanism for diagnosing person's hair of health condition. Nakajima and Sasaki [23] proposed an automatic health monitoring system which two subjects had thinning hair due to aging. In their work, hair whirl was recognized by the close circle. Lee et al. [24] developed and manufactured a carbon nanotube/adhesive polydimethylsiloxane-based electroencephalograph (EEG) electrode which EEG signals can be read and recorded from the hairy scalp.

4. Machine-Learning Techniques for Diagnosing and Analyzing Hairy Scalps

Figure 3 shows the system architecture of the intelligent scalp detection system (ISDS) [25]. The ISDS consists of a scalp detector, an app running on a tablet, machine-leaning techniques [2,26], and a cloud management platform. The scalp detector will be connected with the tablet through a Wi-Fi wireless network. Thus, a scalp photo can be captured via the scalp detector. The scalp photo will be taken by the scalp detector, and the recognized result of the scalp will also be sent and displayed to the tablet. Then, we can obtain quantitative data on scalps. Furthermore, based on ISDS, we will compare three state-of-the-art machine-learning methods using scalp image data: deep learning, BOW [27] with machine-learning classifiers, and PHOG [28] with machine-learning classifiers.

A pre-trained model and transfer learning are included in the deep-learning method as shown Figure 4. The ImageNet-VGG-f model [29] is a pre-trained model that has been trained using 1,200,000 images. The results of this pre-trained model are used as the initial parameters with our scale image data set to perform the fine-tuning function, which can reduce the training time compared with training all images and increases the accuracy compared with choosing random initializations.

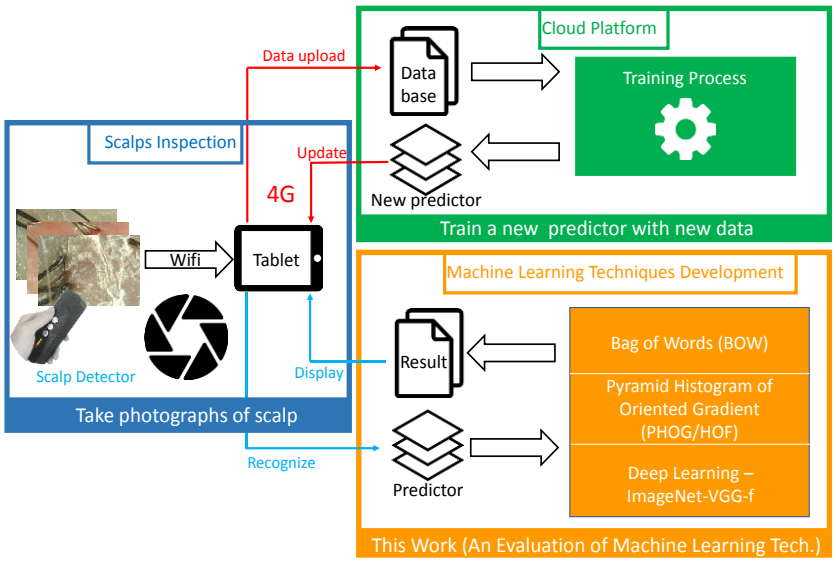


Figure 3. System architecture of the intelligent scalp detection system (ISDS).

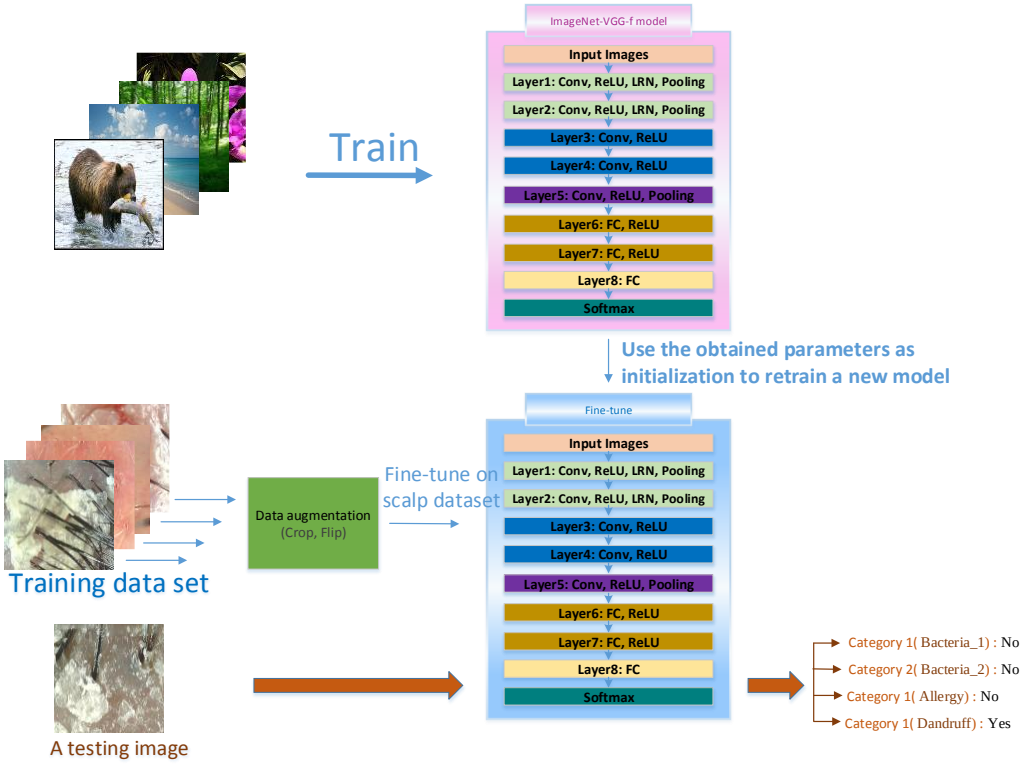


Figure 4. Training and testing of the deep-learning structure.

Figure 5 shows the second method, which is the combination of BOW and SVM. For the BOW, both the training and testing image features are obtained via the SIFT [30] method. Then, we use the K-means [31] method based on SIFT features to train the images to create a codebook. Histograms [32] are built according to the codebook produced for each test image.

Finally, those histograms are used to train a SVM classifier, and the SVM classifier is used to predict the test image. The third method, which is shown in Figure 6, used a HOG algorithm to obtain the training and testing image features. This method trained a SVM classifier based on the HOG features of the training images.

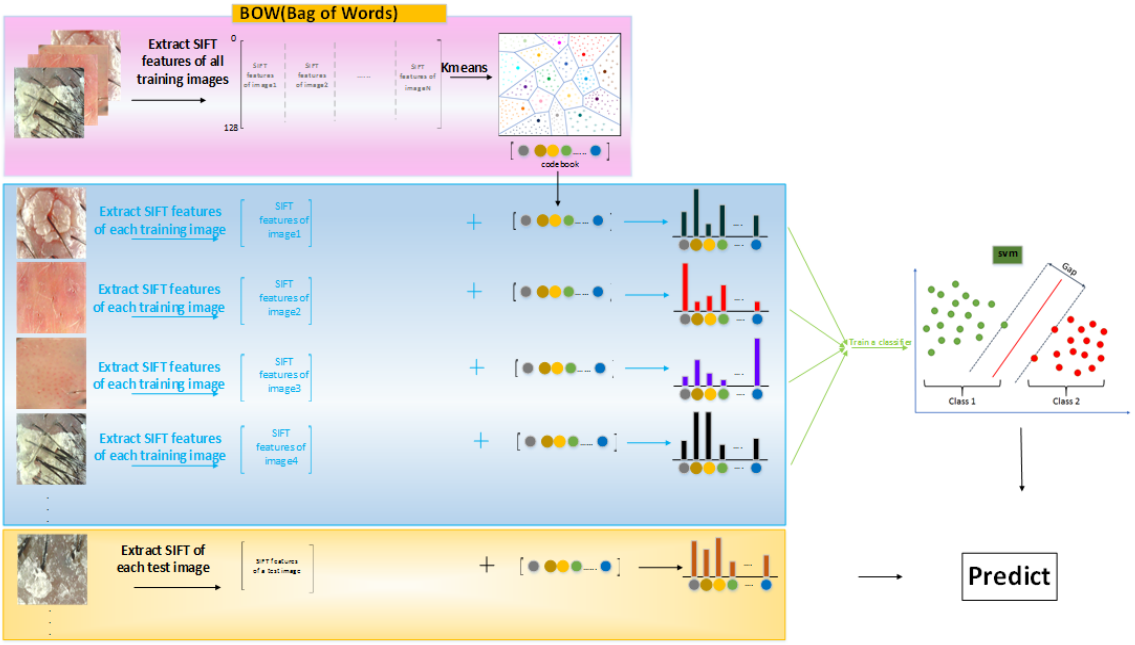


Figure 5. Bag of Words (BOW) with the support vector machine (SVM) method.

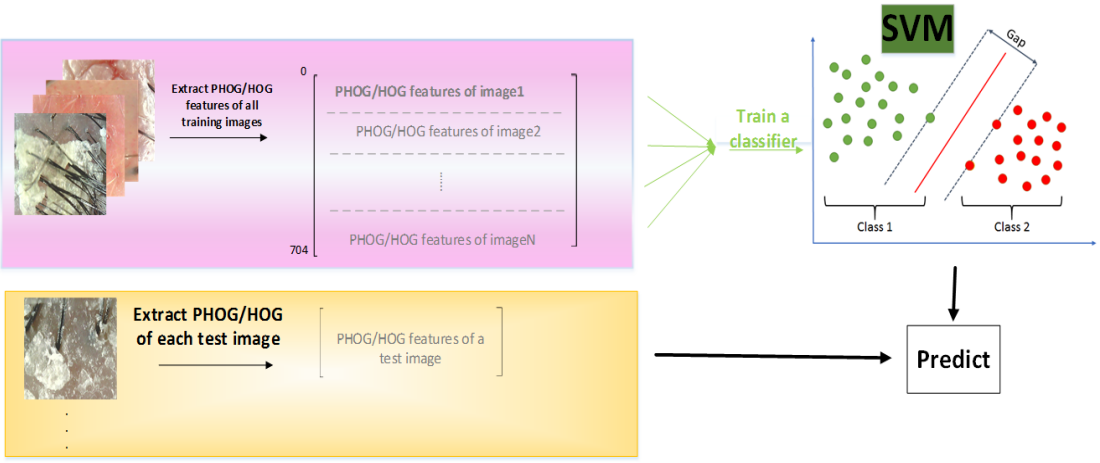


Figure 6. Pyramid histogram of oriented gradients (PHOG)/histogram of oriented gradients (HOG) with the SVM method.

4.1. Deep Learning

Deep learning has become a popular method for image processing and enables automatic feature extraction as a type of feature learning. The process is improved by replacing feature engineering that requires the analysis of knowledgeable and experienced specialists.

The three general steps performed to complete training on the deep-learning framework include defining a network structure, defining a learning target and using a numerical method. The first step is to identify a network structure to choose several possible functions. With a proper network structure, an efficient deep-learning model can be built through the training process. The second step is to define a learning target by choosing objective functions, such as the mean square error (MSE) and cross entropy.

Finally, during the training process, we use numerical methods to discover the best combination of parameters, including weights and bias, to reduce the size of the learning target as much as possible. Backpropagation (BP) is usually used for minimizing an objective function. Deep learning is composed

of a set of functions that can be used to describe data. If the proper parameters of a function can be obtained, we can predict the new input data through those functions. In the following section, we describe how the parameters were updated using BP.

4.1.1. Backpropagation (BP)

BP is utilized to identify the suitable parameters for the deep network. We use the partial derivatives ($\partial J/\partial w$ and $\partial J/\partial b$) of the objective function J regarding every weight w and bias b in the deep neural network. Equation (1) shows the objective function.

$$J = \sum \frac{1}{2}(target - output)^2 = \sum \frac{1}{2}(t - o)^2 \tag{1}$$

where t represents the real class of the images and o represents the output of the deep-learning model. The purpose of BP is to minimize J as much as possible by finding the w. Therefore, BP lessens the difference between the real value and the output of the model. To minimize J and obtain all proper weights, we calculate the partial derivatives of J on w in each layer. We use two layers (as shown in Figure 7) to describe the BP process. First, we calculate the partial derivatives of J from Equation (1) with respect to the weight $w^{(2)}$ of the output layer (as given in Equation (2)). This partial derivative is shown in Equation (3).

$$w^{(2)} = \begin{bmatrix} w_{11}^{(2)} \\ w_{21}^{(2)} \\ w_{31}^{(2)} \end{bmatrix} \tag{2}$$

$$\frac{\partial J}{\partial w^{(2)}} = \begin{bmatrix} \frac{\partial J}{w_{11}^{(2)}} \\ \frac{\partial J}{w_{21}^{(2)}} \\ \frac{\partial J}{w_{31}^{(2)}} \end{bmatrix} \tag{3}$$

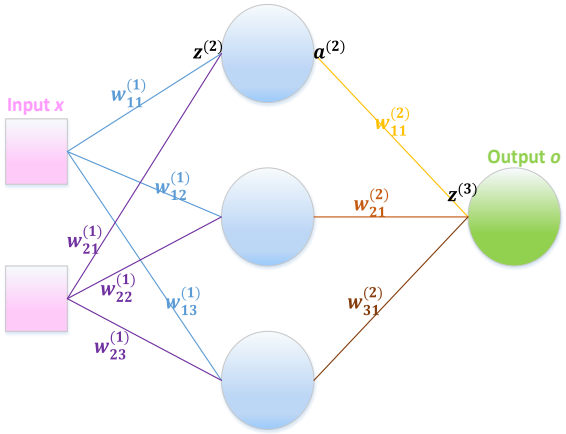


Figure 7. Two-layer neural network.

Target t consists of constants, and the results obtained after performing partial derivatives are shown in Equation (4).

$$\frac{\partial J}{\partial w^{(2)}} = \frac{\partial \sum \frac{1}{2}(t - o)^2}{\partial w^{(2)}} = \sum \frac{\partial \frac{1}{2}(t - o)^2}{\partial w^{(2)}} = -(t - o)\frac{\partial o}{\partial w^{(2)}} \tag{4}$$

The chain rule is utilized in Equation (5).

$$\frac{\partial J}{\partial w^{(2)}} = -(t-o)\frac{\partial o}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial w^{(2)}} \tag{5}$$

Because $o = f(z^{(3)})$, the partial derivatives of o with respect to $z^{(3)}$ are equal to $f'(z^{(3)})$ such that $\frac{\partial o}{\partial z^{(3)}} = f'(z^{(3)})$. Hence, the replacement of $\frac{\partial o}{\partial z^{(3)}}$ by $f'(z^{(3)})$ is shown in Equation (6).

$$\frac{\partial J}{\partial w^{(2)}} = -(t-o)f'(z^{(3)})\frac{\partial z^{(3)}}{\partial w^{(2)}} \tag{6}$$

According to the neural network rule, $z^{(3)}$ (Figure 7) is the result of the multiple $a^{(2)}$ and $w^{(2)}$ such that $z^{(3)} = a^{(2)}w^{(2)}$. Hence, the partial derivative of $z^{(3)}$ regarding the weight $w^{(2)}$ is $a^{(2)}$ as shown in Equation (7).

$$\frac{\partial z^{(3)}}{\partial w^{(2)}} = a^{(2)} \tag{7}$$

For mathematical convenience, replace $-(t-o)f'(z^{(3)})$ with $\delta^{(3)}$ as shown in Equation (8).

$$\delta^{(3)} = -(t-o)f'(z^{(3)}) \tag{8}$$

Therefore, according to Equations (7) and (8), $\frac{\partial J}{\partial w^{(2)}}$ can be represented by Equation (9).

$$\frac{\partial J}{\partial w^{(2)}} = \left(a^{(2)}\right)^T \delta^{(3)} \tag{9}$$

Second, the partial derivatives of J regarding the weight $w^{(1)}$ Equation (10) of the layer before the output layer are represented by Equations (11) and (12). The chain rule is used in Equation (13).

$$w^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(3)} \end{bmatrix} \tag{10}$$

$$\frac{\partial J}{\partial w^{(1)}} = \begin{bmatrix} \frac{\partial J}{w_{11}^{(1)}} & \frac{\partial J}{w_{12}^{(1)}} & \frac{\partial J}{w_{13}^{(1)}} \\ \frac{\partial J}{w_{21}^{(1)}} & \frac{\partial J}{w_{22}^{(1)}} & \frac{\partial J}{w_{23}^{(1)}} \end{bmatrix} \tag{11}$$

$$\frac{\partial J}{\partial w^{(1)}} = \frac{\partial \sum \frac{1}{2}(t-o)^2}{\partial w^{(1)}} = \sum \frac{\partial \frac{1}{2}(t-o)^2}{\partial w^{(1)}} \tag{12}$$

$$\frac{\partial J}{\partial w^{(1)}} = -(t-o)\frac{\partial o}{\partial w^{(1)}} = -(t-o)\frac{\partial o}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial w^{(1)}} \tag{13}$$

Based on $\frac{\partial o}{\partial z^{(3)}} = f'(z^{(3)})$ and Equation (8), $\frac{\partial J}{\partial w^{(1)}}$ can simply be represented by Equation (14).

$$\frac{\partial J}{\partial w^{(1)}} = -(t-o)f'(z^{(3)})\frac{\partial z^{(3)}}{\partial w^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial w^{(1)}} \tag{14}$$

Because $\frac{\partial z^{(3)}}{\partial a^{(2)}}$ is equal to $w^{(2)}$, the results of the replacement after performing the chain rule in Equation (14) are shown in Equation (15).

$$\frac{\partial J}{\partial w^{(1)}} = \delta^{(3)}\frac{\partial z^{(3)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial w^{(1)}} = \delta^{(3)}\left(w^{(2)}\right)^T\frac{\partial a^{(2)}}{\partial w^{(1)}} \tag{15}$$

$\frac{\partial a^{(2)}}{\partial w^{(1)}}=\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial w^{(1)}}$ and $\frac{\partial a^{(2)}}{\partial z^{(2)}} = f'(z^{(2)})$, and Equation (16) displays the result of those substitutions.

$$\frac{\partial J}{\partial w^{(1)}} = \delta^{(3)}(w^{(2)})^T\frac{\partial a^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial w^{(1)}} = \delta^{(3)}(w^{(2)})^T f'(z^{(2)})\frac{\partial z^{(2)}}{\partial w^{(1)}} \tag{16}$$

Also, $\frac{\partial z^{(2)}}{\partial w^{(1)}} = x$, and Equation (17) is the result of substituting x for $\frac{\partial z^{(2)}}{\partial w^{(1)}}$.

$$\frac{\partial J}{\partial w^{(1)}} = x^T\delta^{(3)}(w^{(2)})^T f'(z^{(2)}) \tag{17}$$

Equation (18) shows the short representation after setting $\delta^{(2)}=\delta^{(3)}(w^{(2)})^T f'(z^{(2)})$.

$$\frac{\partial J}{\partial w^{(1)}} = x^T\delta^{(2)} \tag{18}$$

For the partial derivatives of J, the biases $b^{(1)}$ and $b^{(2)}$ are shown in Equation (19).

$$\begin{cases} \frac{\partial J}{\partial b^{(1)}} = \delta^{(2)} \\ \frac{\partial J}{\partial b^{(2)}} = \delta^{(3)} \end{cases} \tag{19}$$

Finally, we use Equations (20) and (21) to update the parameters, including the weights and bias.

$$w_{ij}^{(l)} = w_{ij}^{(l)} - -\alpha\frac{\partial J(w,b)}{\partial w_{ij}^{(l)}} \tag{20}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha\frac{\partial J(w,b)}{\partial b_i^{(l)}} \tag{21}$$

where ij represents the location of the neuron on the l layer and α is the learning rate.

4.1.2. Convolution

Convolution layers are usually composed of a wide range of filters as shown in Figure 8. Those filters can enhance the features of images. For example, the top of Figure 8 shows the pixel representation of a line filter. After convoluting input images with this line filter, a vertical line feature map can be obtained. Through a vertical line feature map, we can extract the line features included in those images. After several layers, the CNN can learn to grab more complex features, such as objects.

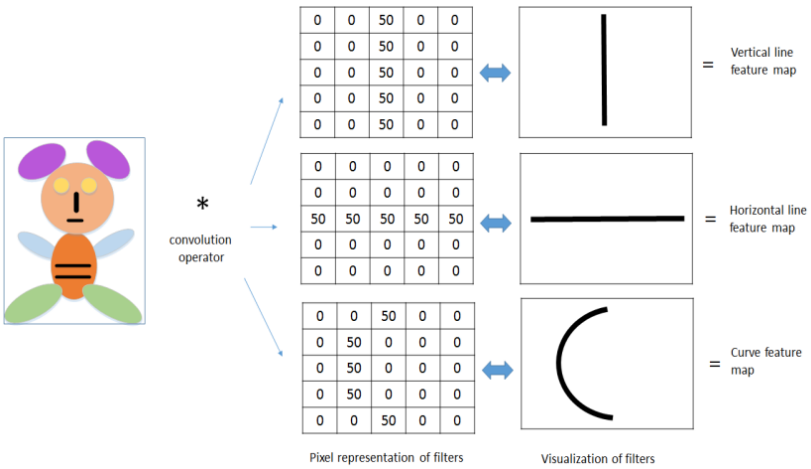


Figure 8. Pixel representation and visualization of filters.

The purpose of training in a CNN is to discover the optimization filters. Through their calculations, testing images can be well represented so that the accuracy of classification can be enhanced. Shared weights and sparse connectivity are two advantages of CNNs that can reduce the number of training parameters.

For shared weights, all neurons in the same hidden layers use the same filter to detect the same feature within an image, such as the line or edge feature. Therefore, different parts of an image use the same filter that includes weights and bias. Sparse connectivity is related to a neuron on the layer being connected to several (not all) neurons of the previous layer.

4.1.3. Rectified Linear Unit (ReLU)

ReLU is a linear activation function and shown in Equation (22). If the input of ReLU, x, is less than 0, then the value of ReLU is zero. If the input of ReLU, x, is larger than 0, then the value of ReLU is still x. Compared with non-linear activation functions, sigmoid and tanh can perform astronomical calculations because both are exponential functions. Most importantly, sigmoid and tanh functions have gradient vanishing problems when implementing BP, which causes missing information.

$$ReLU = max(0, x) \tag{22}$$

4.1.4. Max-Pooling

After performing convolution, the number of parameters is too high to train a classifier, such as a Softmax. Thousands of parameters create an overfitting problem and cause astronomical calculations. As shown in Figure 9, the highest value is obtained from a small area in the previous layer. Hence, Max-pooling is proposed to reduce the number of parameters and prevent overfitting.

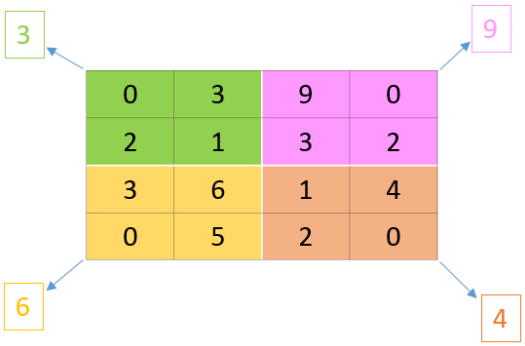


Figure 9. Max-pooling.

4.1.5. Fully Connected Layers (FC)

For the ImageNet-VGG-f model, layers 6 to 8 are fully connected layers (FC). Each neuron of the FC connects to every neuron of the previous layer. Hence, the number of parameters of the FC is large. Layers six and seven include activate functions. High-level features can be calculated from layer five, the convolutional layer, the ReLU and the pooling layer.

The main concept of FC is to transform high-level features into one dimension for classification purposes. Therefore, by using matrix multiplication, FC condenses all dimensions into one vector for Softmax to train a classifier.

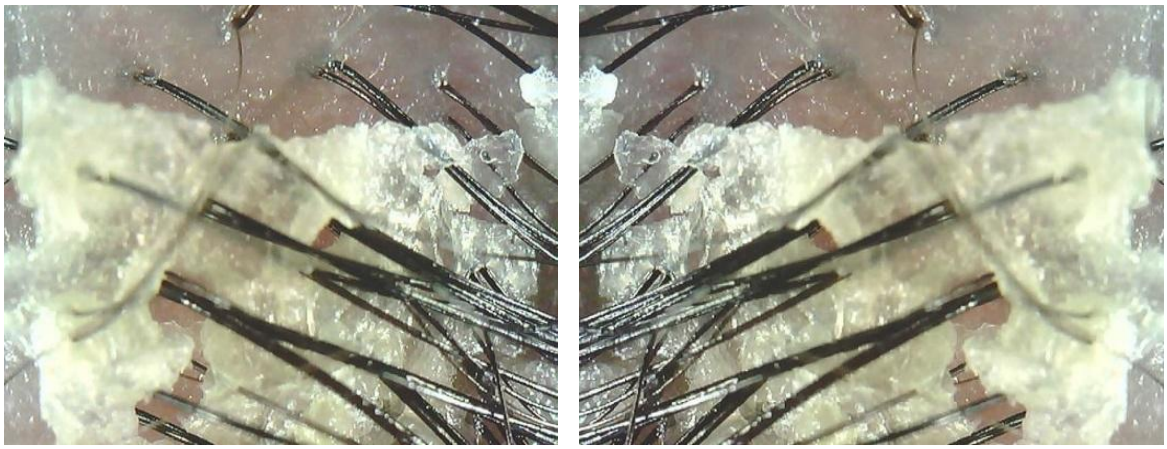
4.1.6. Softmax

The Softmax layer is based on a logical regression to manage multiclass problems. Therefore, this layer is also called a multinomial logistic regression. In this research, four classes are used: bacteria type 1, bacteria type 2, allergy and dandruff. The output of Softmax presents four probabilities of belonging to a class, and the sum of the four probabilities is equal to 1.

4.1.7. Data Augmentation

Before performing fine tuning through the ImageNet-VGG-f pre-trained model, four categories are included with a different number of images in each. Hence, the data augmentation method is applied to equalize the number of images in those four classifications. Flip and Crop are two data augmentation methods.

We use the fliplr function of MATLAB to exchange the columns in order in the horizontal direction. For example, the first column of the image is exchanged with the last a column of the image. Figure 10a shows the original image, and Figure 10b shows the result of flipping the original image. The Crop method crops part of the original images. The data augmentation method can help to reduce the effect of overfitting.



(**a**) (**b**)

Figure 10. (**a**) Scalp before flipping; (**b**) scalp after flipping.

4.2. Bag of Words (BOW)

BOW was originally used to detect words. The general purpose of this method is to obtain visual words from all documents and then calculate the number of occurrences of those visual words based on each document. Subsequently, the occurrences are presented by a feature vector in each document. The flow chart in Figure 11 shows the four key parts of the BOW method: (1) feature detection; (2) feature description; (3) cluster; and (4) histogram.

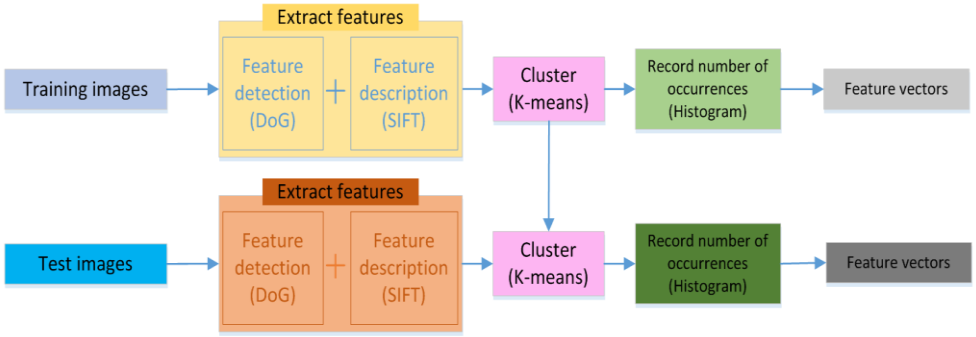


Figure 11. BOW structure.

4.2.1. Feature Detection

Feature detection detects the interesting points of images and can be divided into two categories: global features and local features. For global features, the information contained in an entire image is used, and the most popular global feature is GIST [33].

For local features, the images are divided into many sub-images or segmented according to objects within the images. The feature detection of BOW uses local features to detect the critical points of images. Many methods are used to detect key points, such as Harris–Affine, Hessian–Affine [34], maximally stable extremal regions (MSER) [35], Lowe's difference-of-Gaussian (DOG), edge-based regions (EBRs), intensity-based regions (IBRs) [36], and salient regions [37]. Tuytelaars and Mikolajczyk [38] surveyed many local feature detectors.

4.2.2. Feature Description

After detecting the key points of images, we must accurately describe those key points. In this work, we use VLFeat [39], which combines Lowes DOG and SIFT. By exploiting the features of SIFT, such as its invariance to image scale and rotation, suitable information can be obtained.

Thus, the same SIFT features can be obtained regardless of the rotation and scale of the image. Through SIFT, we can finally obtain a 128-dimension feature vector from each key point. Four steps are performed to obtain the SIFT features: (a) scale-space peak selection; (b) key point localization; (c) orientation assignment; and (d) key point description.

(a) Scale-space Peak Selection

The SIFT algorithm identifies the interesting points in different scale spaces. By multiplying the Gaussian kernel with input images, we can obtain different scale spaces. Equation (23) shows the two-dimensional variable scale space's Gaussian function and Equation (24) shows the scale-space.

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{23}$$

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{24}$$

where $L(x,y,\sigma)$ is the result of the convolution operation with an input image $I(x,y)$ and the two-dimensional space's Gaussian function and σ is the standard deviation. If the value of σ increases, then the image will be blurred, and vice versa. Multiple Gaussian scale spaces can be obtained by using the convolution operation on an original image and varying σ.

Figure 12 shows that the Gaussian pyramid is constructed by different octaves. Figure 13 shows the four octaves of a dandruff image and five standard deviations in each octave. The images in the same octave are the same size but have different standard deviations.

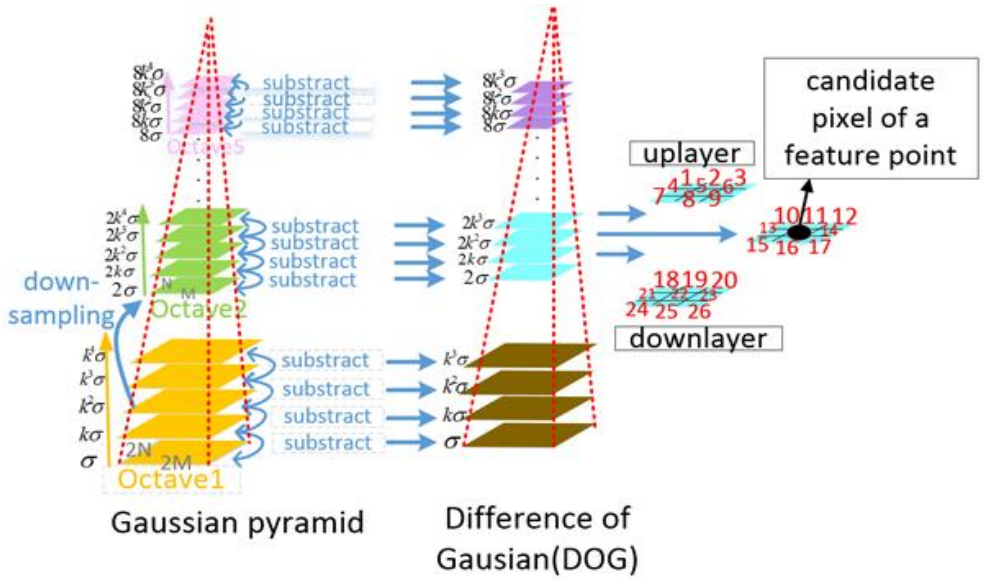


Figure 12. Difference-of-Gaussian (DOG).

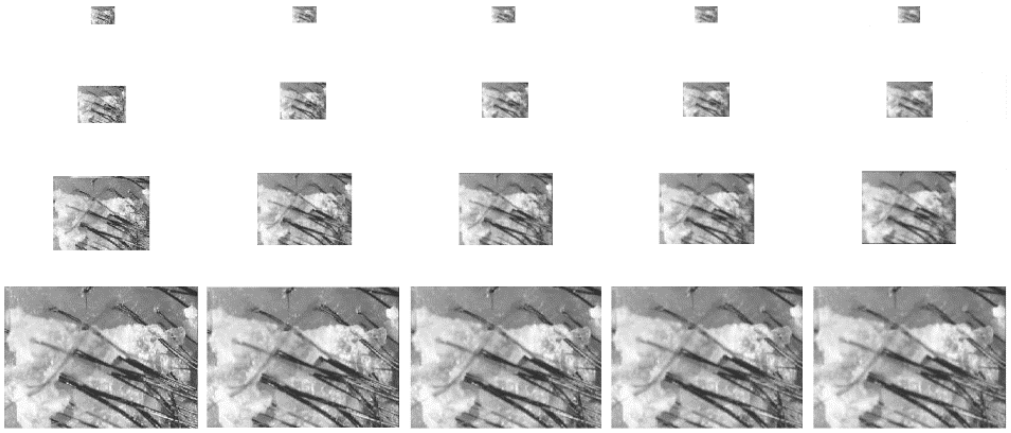


Figure 13. Dandruff image is displayed at four octaves with five standard deviations.

For example, at the bottom of octave1, the σ value is the lowest and the upper layer is k times σ. As σ increases, a variety of scale spaces can be obtained. For different octaves, the next octave is obtained by down-sampling the previous octave. Based on the Gaussian pyramid, the DOG can be generated by subtracting two adjacent images in the same octave.

Comparing 26 neighbor pixels of the candidate pixel (as shown in the right side of Figure 10) can obtain a feature point as the local maxima and minima in the DOG space. The 26 neighbor pixels include the 18 pixels of the up and down layer of DOG and the 8 pixels at the current layer of DOG. To identify stable key points within the scale space, the DOG scale-space is obtained by convoluting different DOG kernels and original images as shown in Equation (25).

$$D(x,y,\sigma) = (G(x,y,k\,\sigma) - G(x,y,\sigma) * I(x,y)) = L(x,y,k\,\sigma) - L(x,y,\sigma) \tag{25}$$

(b) Key Point Localization

Because all local maxima and minima in the DOG space are not feature points, many redundant points must be removed. Brown and Lowe [40] proposed using the Taylor expansion of the scale-space function (as shown in Equation (26)) to increase the accuracy of identifying feature points by removing candidate points that have low contrast.

$$D(x,y,\sigma) = D(x,y,\sigma) + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}X \tag{26}$$

where $x = (x,y,\sigma)^T$ is the offset from the local maxima or minima sampled. Then, the derivative of Equation (26) is applied and set to zero. The accurate location of the local maxima or minima sampled can be obtained as $\hat{x}$ as shown in Equation (27).

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \tag{27}$$

Then, we place Equation (27) into Equation (26), and the result of the first two items is shown in Equation (28).

$$D(\hat{x}) = D(x,y,\sigma) + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \tag{28}$$

Finally, the value of $D(\hat{x})$ at the minima or maxima must be larger than the threshold as shown in Equation (29).

$$|D(\hat{x})| > \text{threshold} \tag{29}$$

If the value of $|D(\hat{x})|$ is less than the threshold, then those points are unstable and can be removed. Removing the interesting points with low contrast is insufficient because of the strong response at the image edges within the difference of the Gaussian function. Two situations were observed: when the principle curvature is large across the edge and when the principle curvature is small on the perpendicular edge. Therefore, Lowe used the 2*2 Hessian matrix to create the principle curvatures. The Hessian matrix is shown in Equation (30).

$$\mathrm{H} = \begin{bmatrix} D_{xx}(x,y) & D_{xy}(x,y) \\ D_{xy}(x,y) & D_{yy}(x,y) \end{bmatrix} \tag{30}$$

where $D_{xx}(x,y)$ is the second-order partial derivative of $D(x,y,\sigma)$ in the x direction at a sample point in DOG. Then, Lowe calculated the individual eigenvalues and only considered the ratio in Equation (31). The numerator of this ratio is the square of the sum of the eigenvalues from the value of the diagonal of H as shown in Equation (32). The denominator of this ratio is the product from the determinant as shown in Equation (33).

$$\frac{\mathrm{Tr}(\mathrm{H})^2}{\mathrm{Det}(\mathrm{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} \tag{31}$$

$$\mathrm{Tr}(\mathrm{H})^2 = D_{xx}(x,y) + D_{yy}(x,y) = \alpha + \beta \tag{32}$$

$$\mathrm{Det}(\mathrm{H}) = D_{xx}(x,y) * D_{yy}(x,y) - (D_{xy}(x,y)^2) = \alpha\beta \tag{33}$$

where α is the largest of the eigenvalues and β is the smallest of the eigenvalues. After setting α to $\gamma\beta$, the ratio is as shown in Equation (34).

$$\frac{\mathrm{Tr}(\mathrm{H})^2}{\mathrm{Det}(\mathrm{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\gamma\beta+\beta)^2}{\gamma\beta^2} = \frac{(\gamma+1)^2}{\gamma} \tag{34}$$

The smallest value of $\frac{(\gamma+1)^2}{\gamma}$ is obtained when α and β are the same, and the ratio increases when γ increases. Hence, we check (35) instead of comparing the ratio of principle curvatures that are less than the threshold γ.

Figure 14a,b show the Bacteria_1 image before and after the threshold method. The numbers of the points before and after the threshold are counted in Figure 14a,b as 1198 and 610, respectively.

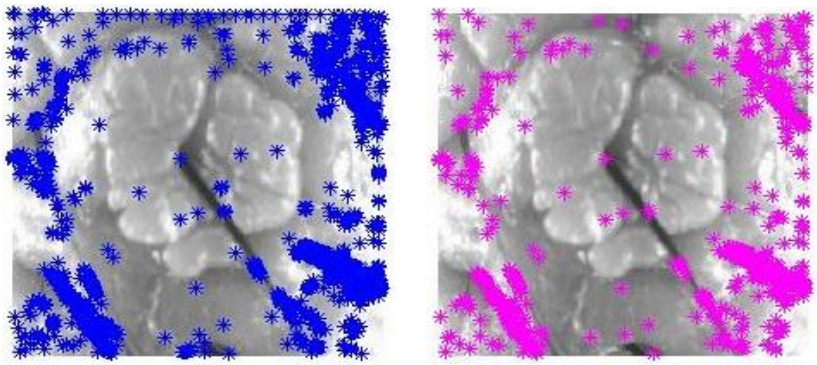
$$\frac{\mathrm{Tr}(\mathrm{H})^2}{\mathrm{Det}(\mathrm{H})} < \frac{(\gamma+1)^2}{\gamma} \tag{35}$$

(c) Orientation Assignment

After finding the locations of key points, we assign an orientation to those key points to obtain the benefits of orientation invariance. By calculating Equations (36) and (37) on adjacent pixels around the key points, we then obtain their gradient magnitudes M_{sift} and orientation θ_{sift}. An orientation histogram with 10 degrees for each bin is built according to this statistic. The main directions of the local gradients are high bins. Except for the highest bin in this orientation histogram, the other key points can also be generated when their bins are more than 80% of the highest bin. Figure 15a shows the gradient magnitude and orientation of a key point. Similarly, Figure 15b shows the gradient magnitude and orientation for all key points of an image.

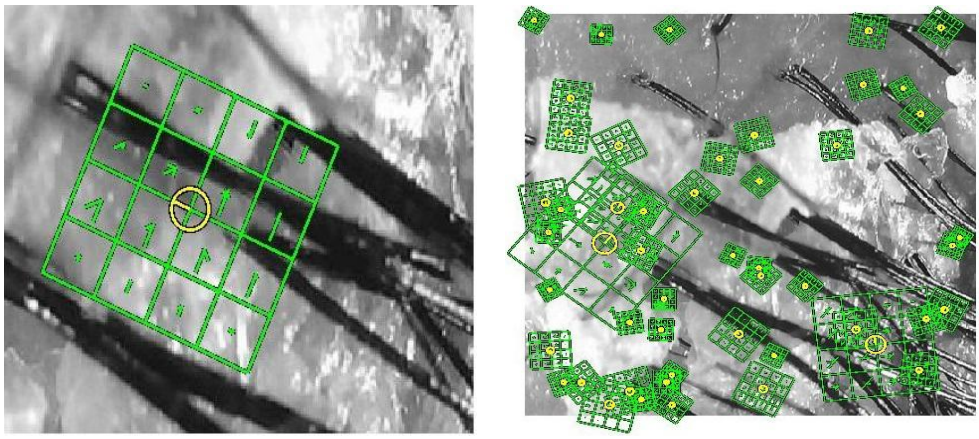
$$M_{sift} = \sqrt{(L_{x+1,y} - L_{x-1,y})^2 + (L_{x,y+1} - L_{x,y-1})^2} \tag{36}$$

$$\theta_{sift} = tan^{-1}((L_{x,y+1} - L_{x,y-1})/(L_{x+1,y} - L_{x-1,y})) \tag{37}$$



(**a**) (**b**)

Figure 14. (**a**) Before the operating threshold method; (**b**) after the operating threshold method.



(**a**) (**b**)

Figure 15. (**a**) Gradient magnitude and orientation of a key point; (**b**) gradient magnitude and orientation of all key points found.

(d) Key Point Description

The key point descriptor primarily describes each key point through a 128-dimension vector. After rotating the main direction of a key point, Lowe calculated the gradient magnitude and orientation of 16*16 adjacent pixels of a key point. The key point as shown in Figure 16a is at the center of the square. The length of each arrowhead represents the gradient magnitude, and the direction of each arrowhead is the orientation.

Then, Lowe divided those 16*16 grids into 4*4 subregions as shown in Figure 16b. Each sub-region presents eight directions, such as 45°, 90°, 135°, 180°, 225°, 270°, 315°, and 360°. Finally, we can obtain a 4*4*8-dimension vector to describe each key point.

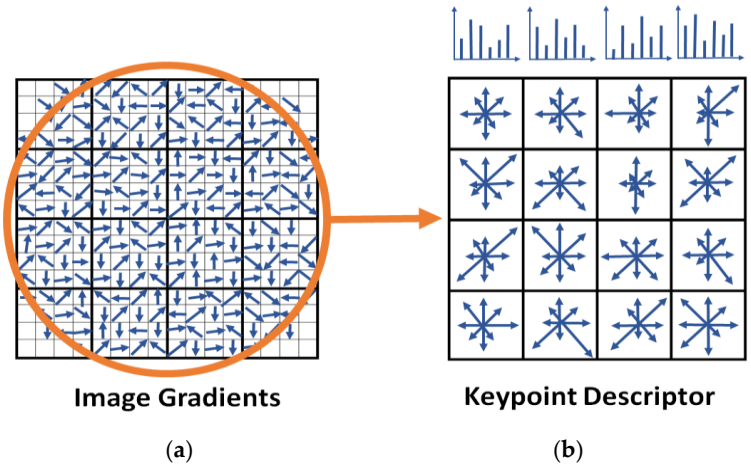


Figure 16. (**a**) Image gradient; (**b**) key point descriptor.

4.2.3. Cluster (K-Means)

K-means clustering is an unsupervised learning method performed to gather similar objects into same groups. Therefore, the minimum of Equation (38) finds the shortest distance for each datum from $x_1 \ldots\ldots x_j$ that belongs to centroid C_i.

$$\text{SSE} = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2 \tag{38}$$

where SSE represents the sum of squares error.

Five steps are performed to achieve K-means clustering:

Step 1: Randomly pick k number of cluster centroids, $\mu_1 \sim \mu_k$.

Step 2: Calculate the closest distance between the remaining data and those k centroids. Then, assign each datum to the nearest cluster centroid.

Step 3: Recalculate each cluster centroid using the meaning of each cluster.

Step 4: Regroup all data according to the new cluster centroids.

Step 5: Repeat Step 4 until certain conditions are met, including a lack of change in centroids, a small change of SSE and no data movement.

4.2.4. Histogram

A histogram calculates the number of occurrences of training images based on the codebook obtained via K-means. Then, we normalize the histograms for the training SVM. For the testing image data set, we calculate the normalized histograms based on the same codebook.

4.3. Histogram of Oriented Gradient (HOG)

Compared with SIFT feature extraction, we also use the global feature extraction method HOG and its extension PHOG. In this research, the VLFeat [39] function vl_hog is used. The essential purpose of HOG is to divide an image into many blocks, and each block contains cells. For each cell, Dalal built a histogram based on the gradient direction and orientation of each pixel.

The combination of histograms of 2*2 cells can represent the descriptor of the image. Then, to improve the performance of HOG, Dalal normalized all cells inside each block according to the intensity of each block, which is called contrast normalization. This contrast normalization process can reduce the effects of shadowing and light changes. The general scheme of HOG is described as follows and shown in Figure 17.

Stage A: Centered gradients of the horizontal and vertical directions are calculated. Through Equations (39) and (40), we can obtain the center horizontal and vertical gradients, respectively, where $[-1\ 0\ 1]$ is the 1D centered-point discrete derivative mask of the horizontal direction and $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ is the same mask in the vertical direction.

$$I_x = I * [-1\ 0\ 1] \tag{39}$$

$$I_y = I * \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{40}$$

Stage B: The distribution of the intensity gradient and the orientation of an image can well represent the local object appearance and shape. Equation (41) is used to obtain the gradient magnitude, and Equation (42) is utilized to obtain the orientation.

$$\left|M_{hog}\right| = \sqrt{I_x^2 + I_y^2} \tag{41}$$

$$\theta_{hog} = \tan^{-1}\frac{I_x}{I_y} \tag{42}$$

Stage C: A 64*128 image is split into blocks. Dalal considered 2*2 close cells (as shown in Figure 15) to represent a block. Each cell consists of four 8*8 pixels. Hence, a total of 105 blocks are within a 64*128 image and a 50% overlap occurs between two blocks.

Stage D: Histograms are built. For this stage, 180 degrees are divided into nine bins of 20 degrees/each. Then, the calculation of the orientation of each bin within a cell produces a histogram. The combination of four histograms of four adjacent cells is a histogram of the block.

Stage E: All histograms are combined. Finally, for an image, 3780 (105(blocks)*4(cells)*9(bins)) features can be obtained.

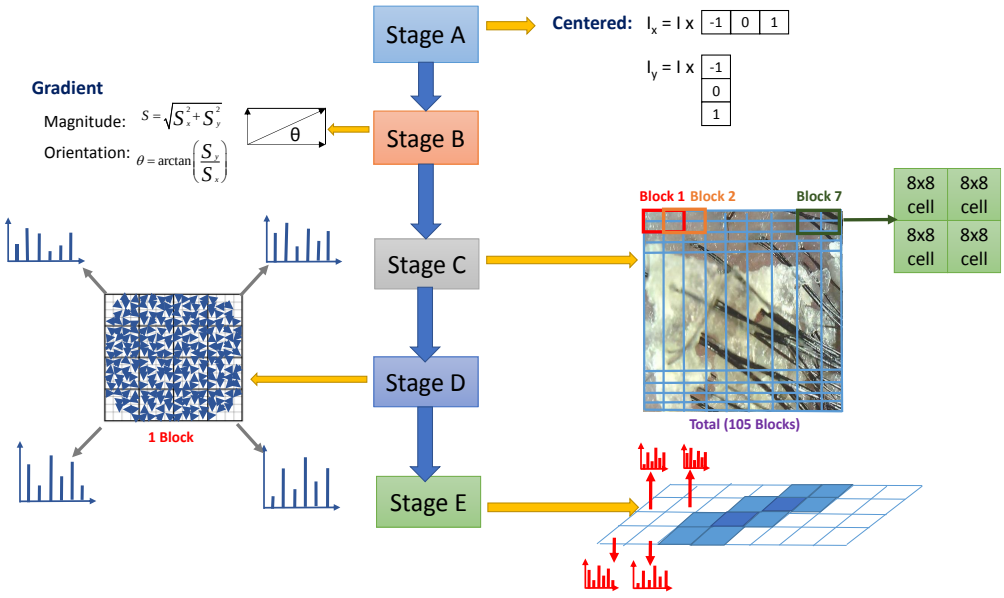


Figure 17. General scheme of HOG.

4.4. Pyramid Histogram of Oriented Gradient (PHOG)

PHOG [41] is an extension of HOG. First, we calculate the different scales of an image. Then, on the same scale, we divide the image into several patches, such as 2*2, 4*4, etc. Subsequently, we calculate the HOF features of those different patches and place all HOF feature results into a one-dimensional array. Finally, we perform the same approach for all other scales of this image and combine the results into the same array. Compared with the HOG method, PHOG can detect features at a variety of scales of an image and can more actively represent the image.

4.5. Machine-Learning Classifiers

In this work, we apply Classification Learner Apps [42], which is a tool included in Matlab. Classification Learner Apps includes a variety of machine-learning classifiers such as SVM, decision tree, linear discriminant analysis (LDA), KNN, and ensemble learning.

4.5.1. Support Vector Machine (SVM)

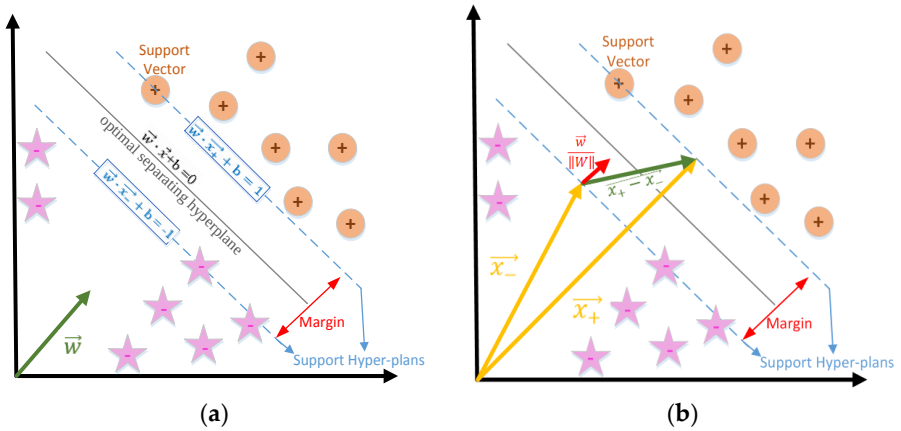
SVM is a classification algorithm proposed by Vapnik [27], and it is based on statistical learning theory. Classification plays a significant role in data mining. The main purpose of the SVM method is to train a classifier via supervised learning. Then, the data can be classified using this model. The purpose of SVM is to calculate the optimum separation hyperplane whose margin is the largest distance from the closest data as shown in Figure 18a. As we may obtain many hyperplanes, the optimum separation hyperplane Equation (43) can reduce the effect of noise and decrease the possibility of overfitting. The largest margin separates the pluses from the minuses as much as possible.

$$\vec{w} \cdot \vec{x} + b = 0 \tag{43}$$

where $\vec{w}$ is the perpendicular to the optimum separation hyperplane and is the inner product. A number of $\vec{w}$ are perpendicular to the optimum separation hyperplane because $\vec{w}$ can be any length and insufficient constraints are available to fix a particular b or a particular $\vec{w}$. Hence, additional constraints are required to calculate b and $\vec{w}$. The super hyperplanes with + labeled data and the super hyperplanes with − labeled data are presented in Equations (44) and (45), respectively.

$$\vec{w} \cdot \vec{x_+} + b = 1 \tag{44}$$

$$\vec{w} \cdot \vec{x_-} + b = -1 \tag{45}$$



(**a**) (**b**)

Figure 18. (**a**) Margin between two support hyperplanes; (**b**) margin is the projection of the different vector on the normal unit.

Hence, if the data belong to the $y_i = 1$ class, then x_+ data can be described by Equation (46). Similarly, if the data are located in the $y_i = -1$ class, then x_- data can be described by Equation (47).

$$\vec{w}\cdot\vec{x_+} + b \geq 1, \qquad \& \text{ if } y_i = 1 \tag{46}$$

$$\vec{w}\cdot\vec{x_-} + b \leq -1, \qquad \& \text{ if } y_i = -1 \tag{47}$$

Then, multiply $y_i = 1$ by Equation (46) and multiply $y_i = -1$ by Equation (47) to obtain the result shown in Equation (48).

$$\begin{cases} y_i(\vec{w}\cdot\vec{x_+} + b) \geq 1, & y_i = 1 \quad \text{For} + \text{samples} \\ y_i(\vec{w}\cdot\vec{x_-} + b) \geq 1, & y_i = -1 \quad \text{For} - \text{samples} \end{cases} => y_i(\vec{w}\cdot\vec{x_i} + \mathrm{b}) - 1 \geq 0 \tag{48}$$

where $y_i(\vec{w}\cdot\vec{x_i} + b) - 1 = 0$ for x_i in the support hyperplanes. The margin is the projection of the different vector $\vec{x_+} - \vec{x_-}$ on the normal unit $\frac{\vec{w}}{\|W\|}$, which is normal to the optimum separation hyperplane as shown in Figure 18b.

Therefore, the dot product of this different vector with the normal unit is the distance between those two support hyperplanes as shown in Equation (49), where $\vec{x_-}$ in Figure 18b is a vector originating from Equation (45) and $\vec{x_+}$ is a vector originating from Equation (44). According to Equations (44) and (45), $w\vec{x_+} = 1 - b$ and $(\vec{w}\vec{x_-}) = -1 - b$ are obtained. Finally, by inserting the right side of those two equalities into Equation (49), the margin is $\frac{2}{\|W\|}$.

$$\text{Margin} = (\vec{x_+} - \vec{x_-})\cdot\frac{\vec{w}}{\|W\|} = \frac{\vec{x_+}\vec{w} - (\vec{x_-}\vec{w})}{\|W\|} = \frac{1 - b - (-1 - b)}{\|W\|} = \frac{2}{\|W\|} \tag{49}$$

For the sake of mathematical convenience, maximizing the margin $\frac{2}{\|W\|}$ is equivalent to maximizing $\frac{1}{\|W\|}$. Also, minimizing $\|W\|$ is equivalent to minimizing $\frac{1}{2}\|W\|^2$ as shown in Equation (50).

$$\text{Max}\frac{2}{\|W\|} \approx \text{Max}\ \frac{1}{\|W\|} \approx \text{Min}\ \|W\| \approx \text{Min}\ \frac{1}{2}\ \|W\|^2 \tag{50}$$

Hence, the solution that identifies the optimum separation hyperplane can solve Equation (51).

$$\begin{cases} \text{Minimize}: \ \frac{1}{2}\ \|W\|^2, \\ \text{Subject to } y_i(\vec{w}\cdot\vec{x_i} + b) - 1 \geq 0 \end{cases} \tag{51}$$

The solution for the extremum of a function with constraints can use Lagrange multipliers. The benefit of using Lagrange multipliers is to maximize or minimize the equation without considering the constraints. Using the Lagrange multiplier method, Equation (51) can be transformed into the quadratic equation shown in Equation (52).

$$L = \frac{1}{2}\|W\|^2 - \sum \alpha_i \left[y_i(\vec{w}\cdot\vec{x_i} + b) - 1 \right] \tag{52}$$

where α_i is the Lagrange multiplier.

By calculating the derivatives of L and setting the results to zero, the extremum of equation L are identified. Equations (53) and (54) show the derivative of L with respect to w and b, respectively. The result of Equation (53) shows that vector w is the linear sum of all or certain samples.

$$\frac{\partial L}{\partial w} = 0 => \ \vec{w} - \sum \alpha_i y_i \ \vec{x_i} = 0 => \vec{w} = \sum \alpha_i y_i \ \vec{x_i} \tag{53}$$

$$\frac{\partial L}{\partial b} = 0 => -\sum \alpha_i y_i = 0 => \sum \alpha_i y_i = 0 \quad (54)$$

Equation (55) shows the results of plugging the w expression Equation (53) into L in Equation (52).

$$L = \frac{1}{2}(\sum \alpha_i y_i \vec{x_i}) \cdot (\sum \alpha_j y_j \vec{x_j}) - (\sum \alpha_i y_i \vec{x_i}) \cdot (\sum \alpha_j y_j \vec{x_j}) - \sum \alpha_i y_i \cdot b + \sum \alpha_i \quad (55)$$

L can be expressed in Equation (56) because $\sum \alpha_i y_i = 0$.

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i \, y_j \vec{x_i} \cdot \vec{x_j} \quad (56)$$

The objective is to identify the maxima of expression L, and this maximization process depends on x sample vectors. The optimization depends on only the dot product of pairs of samples $\vec{x_i} \cdot \vec{x_j}$.

4.5.2. Decision Tree

A decision tree is a supervised machine-learning model with simple process intuition and high execution efficiency. It is suitable for the prediction of classification and regression data types. Compared with other machine-learning models, the execution speed is a major advantage.

In addition, one feature of the decision tree is that each decision stage is fairly clear (YES or NO). In contrast, logistic regression and SVMs are similar to black boxes. It is difficult for us to predict or understand their internal complexities and operational details. Additionally, the decision tree has provided instructions for us to actually simulate and draw a decision-making process from the root, to each leaf, and to the final node.

4.5.3. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a classification approach in which the data of different classes can be generated based on different Gaussian distributions. The LDA-based classification approach attempts to find a linear combination of the characteristics of two types of objects or events to be able to characterize or distinguish the characteristics or events. The resulting combination can be used as a linear classifier or, more commonly, for dimensionality reduction for subsequent classifications.

4.5.4. k-Nearest Neighbor Algorithm (K-NN)

The k-Nearest Neighbor algorithm (K-NN) is a statistical method for classification and regression. In both cases, the input contains the k closest training samples in the feature space as follows. In the K-NN classification, the output is a classification group. The classification of an object is determined by a "majority vote" of its neighbors, and the most common categories in the k nearest neighbors (k is a positive integer, typically small) determine the category assigned to the object. If $k = 1$, the object's category is directly given by the nearest node. In K-NN regression, the output is the attribute value of the object. This value is the average of the values of its k nearest neighbors.

Therefore, the K-NN algorithm uses the vector space model to classify the concepts into similar class cases. The similarity between the cases is high, and it is possible to evaluate the possible classifications of the unknown class cases by calculating the similarity to the known class cases.

4.5.5. Ensemble Learning

A simple understanding of ensemble learning refers to the use of multiple classifiers to predict data sets, thereby improving the generalization capabilities of the overall classifier. We use the classification problem as an explanation. The classification problem refers to the use of some sort of rule for classification. The problem is in finding a certain function. The idea of ensemble learning can generally be understood in this manner: when classifying new data instances, a plurality of classifiers are trained and the classification results of these classifiers are combined (for example, voting) to determine the classification results to obtain better results and to improve the generalization capabilities of the classifier using multiple decision makers together to determine the classification of an instance.

5. Measurements and Experimental Results

In this work, we use a 200× magnification camera to take scalp images as shown in Figure 19. The scalp examples are sorted into four groups: bacteria type 1, bacteria type 2, allergy, and dandruff groups as shown in Figure 20a–d, respectively.

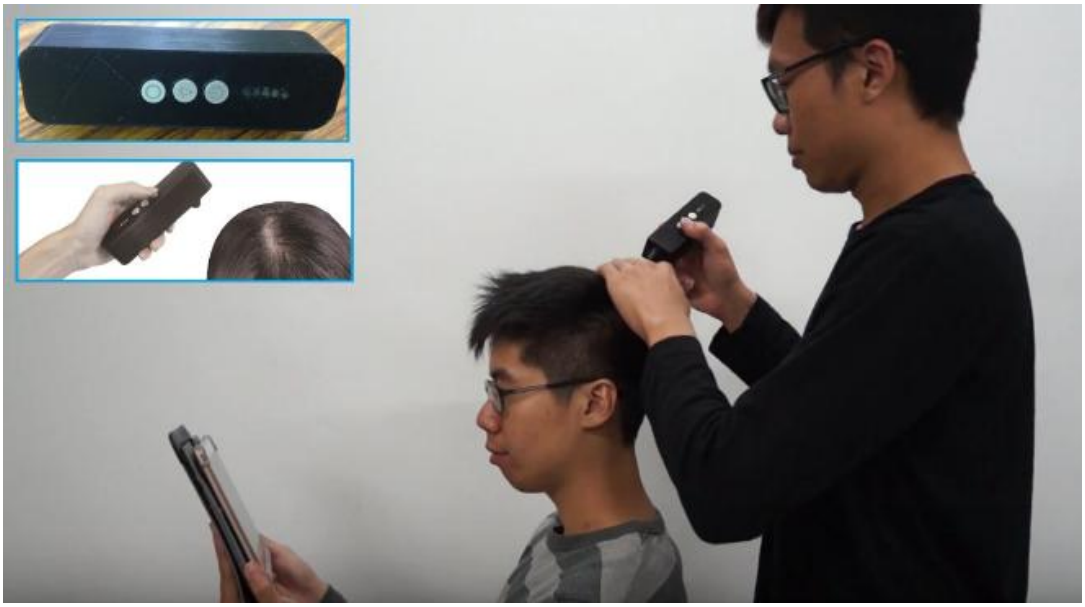


Figure 19. Taking scalp images using a 200× magnification camera.

Bacteria_1 is the condition in which the skin on the scalp presents blisters or boils. Bacteria type 2 is the condition in which the skin on the scalp has many red spots. An allergy scalp problem usually has a considerable number of red patches where the hair root attaches. Dandruff is caused by the shedding of dead scalp skin cells.

For each group, we use an 8:2 rate to divide them into training data and testing data. Thus, 44 images are used for testing and 176 images are used for training. A variety of sizes is observed within those images. The largest image is 2000*15,000, whereas the smallest one is 186*63. Each group contains 220 test images and 880 training images.

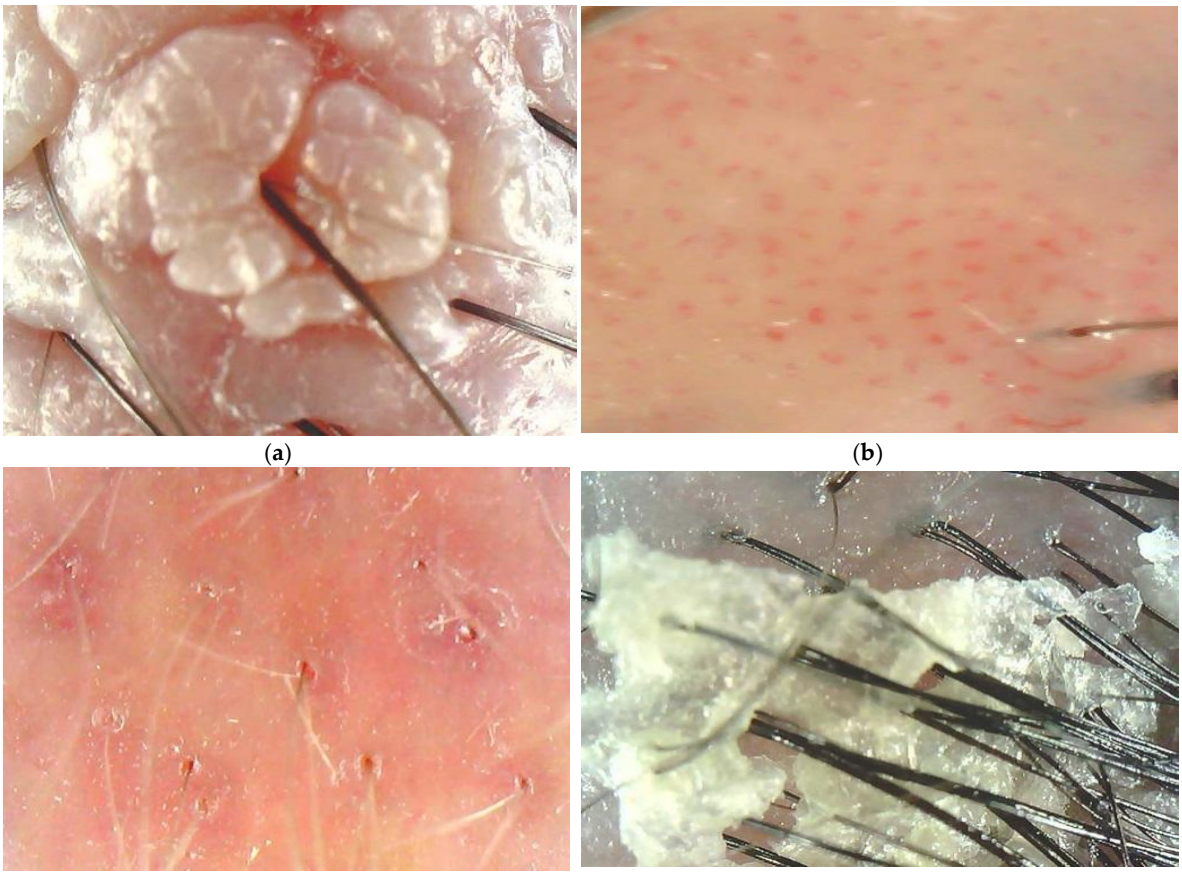


Figure 20. (**a**) Bacteria type 1 image; (**b**) bacteria type 2 image; (**c**) allergy image; (**d**) dandruff image.

5.1. Experimental Results of Deep Learning

As shown in Table 1, the accuracy increases as the learning rate decreases. The three learning rates set in this research are 10^{-4}, 10^{-5} and 10^{-6}.

Table 1. Time for deep learning of training and testing data (seconds).

Learning Rate	Spending on Training Time (s)	Spending on Testing Time (s)	Accuracy
1×10^{-4}	52,798.313	19.36	89.77%
1×10^{-5}	56,821.985	20.826	88.06%
1×10^{-6}	55,571.778	19.451	78.40%

The accuracy assesses the number of correctly predicted images from the deep-learning model with respect to the total number of images. The validation (Val) shown in Figure 21 represents the validation error based on the validation data set. The validation set has been used to estimate the predictive error of the selected model. Similarly, training in Figure 21 represents the training errors created from the training data sets. When the correct label of a testing image is not within the five highest possible labels that the model predicts for this testing image, the top 5 err will be high. Similarly, the top 1 err indicates that the correct label of a testing image is not the label that the model predicts for this testing image. A smaller top 1 err and top 5 err indicates better results. As shown in Table 1, the accuracy increases when the learning rate decreases from 10^{-6} to 10^{-4}.

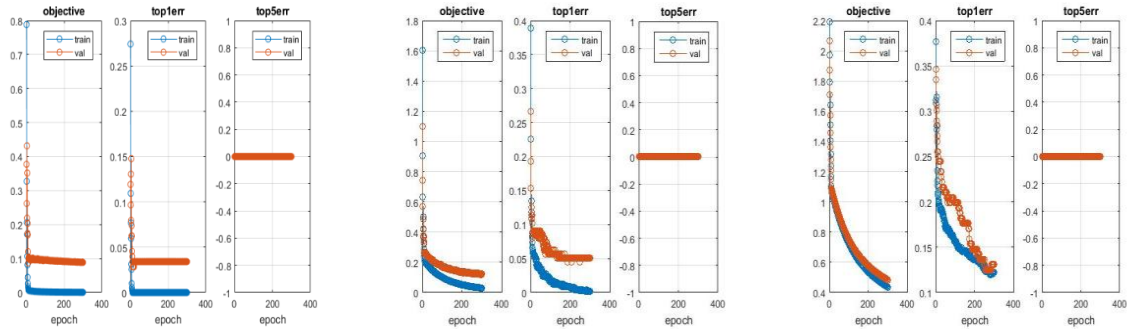


Figure 21. (**a**) Learning rate is 1×10^{-4}; (**b**) learning rate is 1×10^{-5}; (**c**) learning rate is 1×10^{-6}.

5.2. Experimental Results of BOW with Machine-Learning Classifiers

Table 2 shows the time spent on obtaining the SIFT features, calculating K-means and generating histograms for the training data and obtaining the SIFT features and calculating the histograms for the testing data. This discrepancy is because the training data used the K-means results for the training data to create its own histogram.

Table 2. Time for BOW with the training and testing data (in seconds).

Different Centers (Using K-Means Method)	Time to Obtain SIFT Features for Training Data	Time to Obtain SIFT Features for Training Data	Time to Calculate Histograms for Training Data	Time to Calculate Histograms for Training Data
10 centers	472,358.918	2157.486	4253.557	834.538
50 centers	440,101.268	2415.997	4533.5	908.185
100 centers	526,735.405	2854.442	4851.46	963.334
300 centers	481,344.417	3307.601	4791.243	917.323
500 centers	444,823.379	4040.969	4748.997	874.902

The experimental results show that the time spent obtaining the SIFT features from the training data does not increase as the number of centers chosen for the K-means method increases. However, the time spent on the other operations increases as the number of centers increases.

Table 3 shows the accuracies based on different numbers of centers, including 10, 50, 100, 300 and 500. As shown in Table 3, the best accuracy of all scenarios was 80.5% and was achieved with SVM. In Table 3, five machine-learning methods, decision tree, LDA, SVM, K-NN and ensemble learning, are implemented using the features resulting from BOW based on different types of centers selected when using K-means. In Table 3, for those five machine-learning methods, the accuracies increased when selecting 10 centers, 50 centers and 100 centers when using the K-means method. The highest accuracy was achieved when using SVM.

Table 3. Accuracy of BOW based on different machine-learning classifiers.

Classification Learners	10 Centers	50 Centers	100 Centers	300 Centers	500 Centers
Decision Tree	53.8%	59.7%	62.1%	60.7%	60.9%
Linear Discriminant Analysis (LDA)	55.0%	64.2%	67.0%	59.7%	33.9%
Support vector machine (SVM)	68.9%	75.9%	77.4%	80.5%	80.0%
K-Nearest Neighbor (K-NN)	67.9%	74.9%	76.3%	74.7%	76.4%
Ensemble Learning	67.9%	75.4%	78.1%	75.9%	77.3%

5.3. Experimental Results of PHOG/HOG with SVM

Compared with the deep learning and BOW methods, the accuracies of PHOG and HOG are far lower (less than 45%) as shown in Table 4.

Table 4. Training time, testing time and accuracy for PHOG and HOG.

	Training Time	Testing Time	Accuracy without Normalization	Accuracy with Normalization
PHOG	6715.337	1517.237	39.77% (70/176)	44.31% (78/176)
HOG	106,267.868	6294.642	34.65% (61/176)	37.5% (66/176)

Although PHOG is an extension of HOG, the time required to generate PHOG results is far less than that of HOG. Also, the accuracy of applying PHOG is better than HOG, including for the results with and without normalization.

The accuracy of PHOG without normalization is approximately 39.77%, which is 5% higher than that of HOG. Furthermore, after normalization, the accuracy of PHOG is approximately 44%, which is approximately 7% higher than that of HOG. In addition, the time spent on PHOG is 15 times higher than that of HOG. As seen in Table 5, the greatest accuracy was achieved using SVM based on the PHOG features. Compared to the BOG features, when using the same five machine-learning methods, the accuracy of the PHOG features is lower.

Table 5. Accuracy of PHOG based on different machine-learning classifiers.

Classification Learners	PHOG
Decision Tree	39.3%
LDA	33.2%
SVM	53.0%
K-NN	41.3%
Ensemble	50.3%

5.4. Summary

As shown in Table 6, deep learning has the highest accuracy at 89.77%, and the lowest algorithm (PHOG) is at 53.4%. In terms of the time spent on the training and testing data, the PHOG is the quickest algorithm and the BOW is the slowest algorithm, which require 8232.574 and 447,958.95 s, respectively.

Table 6. Running time and accuracy comparisons among the four methods.

Methods	Time Spent	Accuracy
Deep Learning	**(Second Fastest)** 52,817.67	**(Best)** 89.77%
BOW	**(Slowest)** 447,959	**(Worst)** 80.50%
PHOG	**(Faster)** 8232.574	**(Second Best)** 53.30%

6. Conclusions and Future Works

6.1. Conclusions

In this paper, we analyzed scalp images using machine-learning algorithms, including deep learning, BOW with machine-learning classifiers and HOG/PHOG with machine-learning classifiers. The results show the good performance of deep learning, which presented an accuracy rate of 89.77%.

This value is far higher than that achieved by the other two combination methods and higher than the human inspection accuracy of 70.2% of human inspection.

6.2. Future Works

In the future, we hope to use machine-learning algorithms in scalp detectors to automatically detect scalp problems. We will continue collecting additional scalp images to help train the deep-learning model and increase the training accuracy. In addition, we will consider evaluating some possible hybrid deep-learning/non-deep learning methodologies [43,44] for suitable application evaluations to continuously optimize the performance of scalp image recognition. Moreover, other image classification applications will also be trained and evaluated.

Author Contributions: W.-C.W., L.-B.C., and W.-J.C. conceived and designed the experiments; L.-B.C. and W.-J.C. surveyed the related works; W.-C.W. performed the experiments; L.-B.C. and W.-J.C. analyzed the data; W.-J.C. contributed materials and tools; W.-C.W. and L.-B.C. wrote the paper.

Acknowledgments: This work was partially supported by the Ministry of Science and Technology (MOST), Taiwan, under grants MOST-106-2632-E-218-002 and MOST 106-2218-E-218-004.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. The 10 Breakthrough Technologies of 2013. MIT Technology Review. 2013. Available online: https://www.technologyreview.com/s/513981/the-10-breakthrough-technologies-of-2013/#comments (accessed on 10 May 2017).
2. Raschka, S. *Python Machine Learning*, 1st ed.; Packet Publishing: Birmingham, UK, 2015.
3. Loussaief, S.; Abdelkrim, A. Machine learning framework for image classification. In Proceedings of the 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications, Zilina, Slovakia, 5–7 July 2017; pp. 58–61.
4. Caltech 101-Pictures of Objects Belonging to 101 Categories, Computational Vision at Clatech. Available online: http://www.vision.caltech.edu/Image_Datasets/Caltech101/ (accessed on 5 June 2017).
5. Ahmed, S.B.; Naz, S.; Razzak, M.I.; Yousaf, R. Deep learning based isolated Arabic scene character recognition. In Proceedings of the 2017 IEEE International Workshop on Arabic Script Analysis and Recognition, Nancy, France, 3–5 April 2017; pp. 46–51.
6. Jagannathan, S.; Desappan, K.; Swami, P.; Mathew, M.; Nagori, S.; Chitnis, K.; Marathe, Y.; Poddar, D.; Narayanan, S.; Jain, A. Efficient object detection and classification on low power embedded systems. In Proceedings of the 2017 IEEE International Conference Consumer Electronics, Las Vegas, NV, USA, 8–10 January 2017; pp. 233–234.
7. Du, L.; Jiang, W.; Zhao, Z.; Su, F. Ego-motion classification for driving vehicle. In Proceedings of the 2017 IEEE Third International Conference on Multimedia Big Data, Laguna Hills, CA, USA, 19–21 April 2017; pp. 276–279.
8. Pop, D.O.; Rogozan, A.; Nashashibi, F.; Bensrhair, A. Pedestrian recognition through different cross-modality deep learning methods. In Proceedings of the 2017 IEEE International Conference on Vehicular Electronics and Safety, Vienna, Austria, 27–28 June 2017; pp. 133–138.
9. Ermushev, S.A.; Balashov, A.G. A complex machine learning technique for ground target detection and classification. *Int. J. Appl. Eng. Res.* **2016**, *11*, 158–161.
10. Zhao, W.; Du, S. Spectral-spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 4544–4554. [CrossRef]
11. Zhong, Z.; Li, J.; Luo, Z.; Chapman, M. Spectral-spatial residual network for hyperspectral image classification: A 3-D deep learning framework. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 847–858. [CrossRef]
12. Qader, S.H.; Dash, J.; Atkinson, P.M.; Rodriguez-Galiano, V. Classification of vegetation type in Iraq using satellite-based phrenological parameters. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 414–424. [CrossRef]

13. Chen, P.-J.; Ding, J.-J.; Hsu, H.-W.; Wang, C.-Y.; Wang, J.-C. Improved convolutional neural network based scene classification using long short-term memory and label relations. In Proceedings of the IEEE International Conference on Multimedia and Expo Workshops, Hong Kong, China, 10–14 July 2017; pp. 429–434.
14. Singhal, V.; Aggarwal, H.K.; Tariyal, S.; Majumdar, A. Discriminative robust deep dictionary learning for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **2017**, *55*, 5274–5283. [CrossRef]
15. Zhang, H.; Zhuang, B.; Liu, Y. Object classification based on 3D points clouds covariance descriptor. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering and IEEE International Conference on Embedded and Ubiquitous Computing, Guangzhou, China, 21–24 July 2017; pp. 234–237.
16. Remez, T.; Litany, O.; Giryes, R.; Bronstein, A.M. Deep class-aware image denosing. In Proceedings of the International Conference on Sampling Theory and Applications (SampTA'17), Allin, Estonia, 3–7 July 2017; pp. 138–142.
17. Yang, B.; Wang, Y.; Li, J. A spiking-timing-based model for classification. In Proceedings of the 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP'16), Chengdu, China, 16–18 December 2016; pp. 99–102.
18. Zhou, W.; Wu, C.; Du, W. Automatic optic disc detection in retinal images via group sparse regularization extreme learning machine. In Proceedings of the 36th Chinese Control Conference, Dalian, China, 26–28 July 2017; pp. 11053–11058.
19. Chen, T.; Lu, S.; Fan, J. S-CNN: Subcategory-aware convolutional networks for object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**. [CrossRef] [PubMed]
20. Wang, X.; Chen, C.; Cheng, Y.; Wang, Z.J. Zero-shot image classification based on deep feature extraction. *IEEE Trans. Cogn. Dev. Syst.* **2017**. [CrossRef]
21. Nasr, S.; Bouallegue, K.; Shoaib, M.; Mekki, H. Face recognition system using bag of features and multi-class SVM for robot applications. In Proceedings of the 2017 International Conference on Control, Automation and Diagnosis (ICCAD'17), Hammamet, Tunisia, 19–21 January 2017; pp. 263–268.
22. Shih, H.-C.; Lin, B.-S. Hair segmentation and counting algorithms in microscopy image. In Proceedings of the 2015 IEEE International Conference on Consumer Electronics (ICCE'15), Las Vegas, NV, USA, 9–12 January 2015; pp. 612–613.
23. Nakajima, K.; Sasaki, K. Personal recognition using head-top image for health-monitoring system in the home. In Proceedings of the 26th Annual International Conference of the IEEE EMBS, San Francisco, CA, USA, 1–5 September 2004; pp. 3147–3150.
24. Lee, S.M.; Kim, J.H.; Park, C.; Hwang, J.-Y.; Hong, J.S.; Lee, K.H.; Lee, S.H. Self-adhesive and capacitive carbon nanotube-based electrode to record electroencephalograph signals from the hairy scalp. *IEEE Trans. Biomed. Eng.* **2016**, *63*, 138–147. [CrossRef] [PubMed]
25. Chen, L.-B.; Hsu, C.-H.; Su, J.-P.; Chang, W.-J.; Hu, W.-W.; Lee, D.-H. A portable wireless scalp inspector and its automatic diagnosis system based on deep learning techniques. In Proceedings of the 16th International Symposium on Advanced Technology, Tokyo, Japan, 1–2 November 2017.
26. Vapnik, V. *The Nature of Statistical Learning Theory*; Springer: New York, NY, USA, 1995.
27. Sivic, J.; Zisserman, A. Video Google: A text retrieval approach to object matching in videos. In Proceedings of the IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; pp. 1470–1477.
28. Bosch, A.; Zisserman, A.; Munoz, X. Representing shape with a spatial pyramid kernel. In Proceedings of the 6th ACM International Conference on Image and Video Retrieval (CIVR'07), Amsterdam, The Netherlands, 9–11 July 2007; pp. 401–408.
29. Chatfield, K.; Somonyan, K.; Vedaldi, A.; Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. In Proceedings of the British Machine Vision Conference (BMVC), Nottingham, UK, 1–5 September 2014.
30. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
31. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137. [CrossRef]
32. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.

33. Oliva, A.; Torralba, A. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **2001**, *42*, 145–175. [CrossRef]
34. Mikolajczyk, K.; Schmid, C. Scale and affine invariant interest point detectors. *Int. J. Comput. Vis.* **2004**, *60*, 63–86. [CrossRef]
35. Matas, J.; Chum, O.; Urban, M.; Pajdla, T. Robust wide baseline stereo from maximally stable extremal regions. In Proceedings of the British Machine Vision Conference (BMVC), Cardiff, UK, 2–5 September 2002; pp. 384–393.
36. Tuytelaars, T.; Van Gool, L. Matching widely separated views based on affine invariant regions. *Int. J. Comput. Vis.* **2004**, *59*, 61–85. [CrossRef]
37. Kadir, T.; Zisserman, A.; Brady, M. An affine invariant salient region detector. In Proceedings of the European Conference on Computer Vision, Prague, Czech Republic, 11–14 May 2004; pp. 404–416.
38. Tuytelaars, T.; Mikolajczyk, K. Local invariant feature detectors: A survey. *Comput. Graph. Vis.* **2008**, *3*, 177–280. [CrossRef]
39. Vedaldi, A.; Fulkerson, B. VLFeat. An Open and Portable Library of Computer Vision Algorithms. 2008. Available online: http://www.vlfeat.org (accessed on 5 June 2017).
40. Brown, M.; Lowe, D.G. Recognising panoramas. In Proceedings of the IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; pp. 1–8.
41. Bosch, A.; Zisserman, A. PHOG Descriptor. Available online: http://www.robots.ox.ac.uk/~vgg/research/caltech/phog.htm (accessed on 15 June 2017).
42. Classification Learner Apps of Matlab. MathWork. Available online: https://www.mathworks.com/help/stats/classification-learner-app.html (accessed on 24 April 2018).
43. Nanni, L.; Ghidoni, S.; Brahnam, S. Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognit.* **2017**, *71*, 158–172. [CrossRef]
44. Nanni, L.; Ghidono, S. How could a subcellular image, or a painting by Van Gogh, be similar to a great white shark or to a pizza? *Pattern Recognit.* **2017**, *85*, 1–7. [CrossRef]