applied sciences

MDPI

Article

Experimenting with Extreme Learning Machine for Biomedical Image Classification

Francesco Mercaldo [1,2,*], Luca Brunese [1], Fabio Martinelli [2], Antonella Santone [1] and Mario Cesarelli [3]

[1] Department of Medicine and Health Sciences "Vincenzo Tiberio", University of Molise, 86100 Campobasso, Italy; luca.brunese@unimol.it (L.B.); antonella.santone@unimol.it (A.S.)
[2] Institute for Informatics and Telematics, National Research Council of Italy, 56121 Pisa, Italy; fabio.martinelli@iit.cnr.it
[3] Department of Engineering, University of Sannio, 82100 Benevento, Italy; mcesarelli@unisannio.it
* Correspondence: francesco.mercaldo@unimol.it

Abstract: Currently, deep learning networks, with particular regard to convolutional neural network models, are typically exploited for biomedical image classification. One of the disadvantages of deep learning is that is extremely expensive to train due to complex data models. Extreme learning machine has recently emerged which, as shown in experimental studies, can produce an acceptable predictive performance in several classification tasks, and at a much lower training cost compared to deep learning networks that are trained by backpropagation. We propose a method devoted to exploring the possibility of considering extreme learning machines for biomedical classification tasks. Binary and multiclass classification in four case studies are considered to demonstrate the effectiveness of extreme learning machine, considering the biomedical images acquired with the dermatoscope and with the blood cell microscope, showing that the extreme learning machine can be successfully applied for biomedical image classification.

Keywords: extreme learning machines; biomedical image classification

Citation: Mercaldo, F.; Brunese, L.; Martinelli, F.; Santone, A.; Cesarelli, M. Experimenting with Extreme Learning Machine for Biomedical Image Classification. *Appl. Sci.* **2023**, *13*, 8558. https://doi.org/10.3390/app13148558

Academic Editors: Arianna Consiglio and Flavio Licciulli

Received: 6 July 2023
Revised: 19 July 2023
Accepted: 21 July 2023
Published: 24 July 2023

1. Introduction and Background

Since the introduction of the backpropagation algorithm, feedforward neural networks have been extensively utilized and embraced by researchers in both academic and industrial settings. These networks are specifically designed to detect errors by propagating information from the output nodes back to the input nodes [1]. The conventional backpropagation algorithm, being a first-order gradient method for parameter optimization, encounters challenges such as slow convergence and the risk of getting stuck in local minima. To address these issues, researchers have introduced various approaches to enhance the efficiency and optimality of training feedforward neural networks. One notable example is the utilization of second-order optimization methods [2], subset selection methods [3], or global optimization methods [4]. While these methods often result in faster training speeds or an improved generalization performance compared to the backpropagation algorithm, the majority of them are still unable to ensure the attainment of a globally optimal solution.

In recent times, there has been a proposal for an approach called extreme learning machine (ELM) to train single hidden layer feedforward neural networks (SLFNs) [5]. ELM introduces a unique approach where the hidden nodes are randomly initialized and remain fixed without any iterative tuning. Interestingly, these hidden nodes in ELM do not necessarily have to resemble neurons. The only parameters that require learning are the connections, specifically the weights between the hidden layer and the output layer. As a result, ELM is formulated as a linear-in-the-parameter model, simplifying the process to solve a linear system. Compared with traditional learning methods for feedforward neural networks, ELM exhibits remarkable efficiency and has a tendency to converge towards a global optimum. Theoretical studies have demonstrated that even

with randomly generated hidden nodes, ELM preserves the universal approximation capability of single-hidden-layer feedforward networks (SLFNs) [6,7]. Utilizing commonly employed activation functions, ELM has the capability to achieve an approximate optimal generalization bound similarly to that of traditional feedforward neural networks where all parameters are learned [8,9]. The superiority of ELM over the traditional feedforward neural network (FNN) algorithms in terms of efficiency and generalization performance has been demonstrated across various problem domains in different fields [7,10]. It is worth noting that ELM is generally much more efficient than support vector machines (SVMs) [11], least square support vector machines [12], and other state-of-the-art algorithms. Empirical investigations have revealed that the generalization capability of ELM is comparable to, and in some cases even superior to, support vector machines (SVMs) and their variants [7,13,14]. Detailed comparisons of ELM and SVM can be found in Huang [15] and Huang, Zhou, et al. [16].

Over the past decade, the theories and applications of ELM have been subject to extensive research. In terms of learning efficiency, the original objectives of ELM encompass three aspects: minimizing human intervention, achieving high learning accuracy, and facilitating a fast learning speed.

In recent times, notable advancements have been made in utilizing feedforward neural networks for the classification of medical images [17]. Feedforward neural networks, particularly deep learning, which is a branch of machine learning focused on studying multiple layers of perception, have gained substantial attention in the realm of image classification. Deep learning aims to develop hierarchical representations of features, distinguishing lower-level characteristics from higher-level ones, thereby enabling the identification of various higher-level features. Image classification—categorizing images into predefined groups—is a significant challenge in computer vision. Various deep learning methods have been employed to address these challenges, such as utilizing multilayer nonlinear data processing, classification, feature selection, transformation, and structure identification. Among these architectures, convolutional neural networks (CNNs) are the prominent choice for the identification, classification, and monitoring of most medical images.

In recent decades, traditional machine learning techniques such as support vector machines (SVMs) were commonly employed for biomedical image classification purposes [18]. Nevertheless, SVMs also possess certain drawbacks, such as time-consuming feature identification and extraction processes, as well as a relatively lower performance compared to several other methods. As a result, researchers in the field of biomedical image classification are now advocating for the utilization of deep learning-based approaches.

While ELMs offer several advantages, they also have some challenges and limitations to consider, including the lack of iterative training and the fact that ELMs are susceptible to overfitting, especially when the number of hidden neurons is much larger than the available training samples. However, with random initialization sensitivity, the performance of ELMs can also be sensitive to the random weight initialization process. It is important to carefully consider these limitations and assess whether ELMs are the appropriate choice for a particular problem. As a matter of fact, although ELMs offer simplicity, computational efficiency, and good performance on many tasks, they may not be the ideal solution for every scenario.

For example, Ali et al. [19] achieved promising results in biomedical image classification by employing supervised machine learning techniques. On the Herlev dataset and the Herlev benchmark dataset, which consist of 1417 cervical cancer cells and encompass seven diagnostic classes (superficial squamous, intermediate squamous, columnar, mild dysplasia, moderate dysplasia, severe dysplasia, and carcinoma in situ), they achieved a classification accuracy of 0.95 with an F-value of 0.94 and a classification accuracy of 0.88 with an F-value of 0.89, respectively. These results were obtained using deep learning methods.

Urushibara et al. [20] conducted a study demonstrating that deep learning techniques can achieve diagnostic performance equivalent to experienced radiologists in diagnosing cervical cancer based on a single T2-weighted image. They utilized a deep learning model

employing convolutional neural networks, specifically the Xception architecture. The model was trained with 488 images from 117 cancer patients and 509 images from 181 non-cancer patients over 50 epochs. The model's performance was evaluated using 60 images each from 60 cancer and 60 non-cancer patients. Additionally, three blinded experienced radiologists independently interpreted the same set of 120 images. The deep learning model achieved a sensitivity of 0.883, specificity of 0.933, and accuracy of 0.908. Comparatively, the radiologists exhibited a sensitivity range of 0.783–0.867, a specificity range of 0.917–0.950, and an accuracy range of 0.867–0.892.

The authors in [21] proposed a method based on deep learning, which involved utilizing fuzzy C-means for MRI segmentation and extracting a set of features using discrete wavelet transform. The deep neural network architecture consisted of seven hidden layers. The results of their approach demonstrated a precision and recall of 0.97 to accurately classify normal brain MRIs and malignant ones.

The researchers in [22] presented a method that leveraged convolutional neural networks consisting of seven hidden layers to detect different grades of brain cancer. They achieved an accuracy of 0.86 in accurately identifying the grades of brain cancer.

Zia et al. [23] addressed the same problem by employing discrete wavelet transform for feature extraction, principal component analysis for feature selection, and support vector machines for the classification task.

Deep learning adoption is associated with several drawbacks. For instance, it typically demands a substantial amount of data to outperform other techniques, making data availability a critical factor: as a matter of fact, there are several techniques that aim to increase the data, for instance, data augmentation exploiting computational fluid dynamics (CFD) simulations . CFD simulations are numerical methods used to analyze and predict the behavior of fluid flows. It is a branch of fluid mechanics that employs computational techniques to solve the governing equations of fluid flow, which are typically partial differential equations. CFD simulations are widely used in engineering, physics, and other scientific fields to study and understand complex fluid flow phenomena. CFD can be used for data augmentation in various ways; for example, in computational fluid dynamics, flow visualizations often generate colorful images representing the flow patterns, velocity fields, pressure distributions, and other related properties. These images can be used as synthetic data to augment existing image datasets. The synthetic images can introduce variations in flow patterns, turbulence, and other fluid properties that may not be readily available in real-world data. Additionally, training deep learning models can be exceedingly expensive due to the complex data models involved. The requirement for expensive GPUs and a considerable number of machines further adds to the cost and resource-intensive nature of deep learning [24]. This increases the cost to the users. We explore the possibility of considering ELM for biomedical image classification. Two biomedical domains (the first one related to the dermatoscopic images while the second one obtained with blood cell microscope) are considered for a total of four different case studies (multiclass and binary classification with dermatoscopic images and multiclass and binary classification with blood cell microscope images) are shown to demonstrate that ELM can be suitable for biomedical image classification. We exploit different networks: the first one is the classic ELM, and the other ones are: weighted extreme learning machines (WELMs), regularized extreme learning machines (RELMs), and multilayer extreme learning machines. Moreover, we compare the models obtained with the ELM model by using different supervised machine learning algorithms: the first one is the AlexNet model [25], a CNN architecture that is really widespread in biomedical image classification [26–28], J48, Random Forest, REPTree, Bayes, and Naive Bayes.

To the best of the authors' knowledge, this paper represents the first attempt to consider four different ELM models in the context of biomedical image classification. As a matter of fact, the current state-of-the-art literature includes several research papers exploring the adoption of ELM with biomedical images (for instance, in reference [29], the authors consider ELM for the classification of benign and malignant cells in breast cancer, and

in [30], ELM is exploited for the diagnosis of COVID-19); however, this represents the first attempt to compare four different ELM architectures with two different bioimages.

This paper proceeds as follows: in the next section, the proposed method for biomedical image classification by exploiting extreme learning machines is presented, the four case studies (belonging to two different biomedical image domains) are presented in Section 4 and, finally, in the last section, conclusions and future research plan are drawn.

2. Background

In this section, we provide preliminary notions about ELM and the several ELM variants we exploit, i.e., weighted extreme learning machines (WELMs), regularized extreme learning machines (RELMs), and multilayer extreme learning machines (MELMs), with the aim of keeping the paper self-contained. We also discuss the differences between the several ELM architectures that we consider.

2.1. ELM

ELMs are a type of machine learning algorithm that fall under the umbrella of feedforward neural networks. ELMs were introduced as a simplified and computationally efficient alternative to traditional neural network training methods.

The key idea behind ELMs is to randomly generate the weights connecting the input layer to the hidden layer, rather than optimizing them through an iterative learning process like gradient descent. This random weight initialization is what makes ELMs "extreme".

Below is a step-by-step overview of how ELMs work:

1. Input Layer: The ELM begins with an input layer that represents the features or attributes of the input data.
2. Hidden Layer: The hidden layer consists of randomly initialized neurons. Each neuron in the hidden layer has its own set of weights connecting it to the input layer.
3. Activation Function: An activation function is applied to the weighted sum of inputs for each neuron in the hidden layer. Common activation functions used in ELMs include sigmoid, tanh, or rectified linear unit (ReLU). The purpose of the activation function is to introduce nonlinearity into the model.
4. Output Layer: The output layer produces the final predictions or classifications. In classification problems, the output layer typically employs a softmax function to produce probabilities for each class. In regression problems, the output layer can use a linear activation function.
5. Random Weight Initialization: In ELMs, the weights connecting the input layer to the hidden layer are randomly generated. These weights are usually drawn from a uniform distribution within a specific range.
6. Calculate Hidden Layer Outputs: The random weights and the input data are multiplied, and the activation function is applied to the resulting weighted sum for each neuron in the hidden layer. This yields the output of the hidden layer.
7. Solve Output Weights: The output weights from the hidden layer to the output layer are calculated using a linear regression technique. This step involves finding the optimal values of the output weights that minimize the difference between the actual outputs and the desired outputs.
8. Prediction: Once the output weights are determined, the ELM can make predictions on new, unseen data by propagating the input through the network and computing the output using the learned weights.

The main advantages of ELMs are their simplicity and computational efficiency. They can be particularly useful when dealing with large-scale datasets or time-constrained applications. However, it is worth noting that ELMs do not provide the same level of interpretability as traditional neural networks trained through iterative methods.

Overall, ELMs offer a different approach to training neural networks by randomly initializing the hidden layer weights and solving the output weights through a linear regression process, making them an attractive option for certain applications.

2.2. WELM

WELMs are an extension of the traditional ELM algorithm. WELMs incorporate a weighting mechanism that assigns different importance levels to each sample in the training set, allowing for more flexible and adaptive learning.

In WELMs, the key difference lies in the inclusion of sample weights during the learning process. The sample weights represent the relative importance or significance of each training example. By assigning higher weights to certain samples, the algorithm can focus more on learning from those instances, giving them more influence on the model's decision making.

In the following an overview of how WELMs work, building upon the steps described for ELMs:

1. Input Layer: The input layer represents the features or attributes of the input data, similar to ELMs.
2. Hidden Layer: The hidden layer consists of randomly initialized neurons. Each neuron has its own set of weights connecting it to the input layer.
3. Activation Function: An activation function is applied to the weighted sum of inputs for each neuron in the hidden layer, introducing nonlinearity.
4. Output Layer: The output layer produces the final predictions or classifications using an appropriate activation function.
5. Random Weight Initialization: The weights connecting the input layer to the hidden layer are randomly generated, just like in ELMs.
6. Calculate Hidden Layer Outputs: The random weights and the input data are multiplied, and the activation function is applied to obtain the output of the hidden layer.
7. Weight Assignment: WELMs introduce the concept of sample weights. Each training example is assigned a weight that represents its importance. These weights can be determined based on various factors, such as data distribution, class imbalance, or domain knowledge.
8. Solve Weighted Output Weights: WELMs modify the process of solving the output weights by considering the sample weights. Instead of a simple linear regression, a weighted least squares approach is employed. The algorithm seeks to minimize the weighted difference between the actual outputs and the desired outputs, giving more weight to the samples with higher importance.
9. Prediction: Once the output weights are obtained, the WELM can make predictions on new data by propagating the input through the network and computing the output using the learned weights.

By incorporating sample weights, WELMs offer greater adaptability and flexibility in learning from data. Samples that are deemed more important contribute more significantly to the model's training, enabling the algorithm to focus on specific instances or address class effectively imbalances.

Weighted extreme learning machines can be particularly beneficial in scenarios where certain data points are more informative or have higher relevance than others. By assigning appropriate weights, the algorithm can effectively adapt to the varying importance levels of training examples and improve the overall performance.

2.3. RELM

RELMs are an extension of the traditional extreme learning machine (ELM) algorithm that incorporates regularization techniques to improve generalization and prevent overfitting. The regularization term is added to the objective function during the training process, encouraging the model to find a balance between fitting the training data and reducing complexity.

Below, an overview of how RELMs work is given, building upon the steps described for ELMs:

1. Input Layer: The input layer represents the features or attributes of the input data, similar to ELMs.
2. Hidden Layer: The hidden layer consists of randomly initialized neurons. Each neuron has its own set of weights connecting it to the input layer.
3. Activation Function: An activation function is applied to the weighted sum of inputs for each neuron in the hidden layer, introducing nonlinearity.
4. Output Layer: The output layer produces the final predictions or classifications, using an appropriate activation function.
5. Random Weight Initialization: The weights connecting the input layer to the hidden layer are randomly generated, as in ELMs.
6. Calculate Hidden Layer Outputs: The random weights and the input data are multiplied, and the activation function is applied to obtain the output of the hidden layer.
7. Regularization Term: RELMs introduce a regularization term to the objective function used during the training process. This term penalizes large weights and adds a constraint on the model's complexity. It helps prevent overfitting, where the model becomes too specialized to the training data and performs poorly on unseen data.
8. Solve Regularized Output Weights: The output weights are determined using a regularized least squares method. This process involves minimizing the sum of the squared errors between the actual outputs and the desired outputs while considering the regularization term.
9. Prediction: Once the output weights are obtained, the RELM can make predictions on new data by propagating the input through the network and computing the output using the learned weights.

The addition of the regularization term in RELMs encourages the model to find a balance between fitting the training data and reducing complexity. By penalizing large weights, RELMs promote simpler models that are less likely to overfit and generalize better to unseen data.

Regularized extreme learning machines are particularly useful when dealing with limited training data or when there is a high risk of overfitting due to complex models. The regularization term allows for better control over the model's complexity, leading to improved generalization and more robust performance.

It is important to note that different types of regularization techniques can be used in RELMs, such as L1 or L2 regularization (also known as Ridge or Lasso regression). The specific choice of regularization method depends on the characteristics of the problem at hand and the desired properties of the learned model.

2.4. MELM

MELMs are an extension of the traditional extreme learning machine (ELM) algorithm that introduces additional hidden layers between the input layer and the output layer. This allows MELMs to capture more complex and abstract representations of the input data.

Here is an overview of how MELMs work, building upon the steps described for ELMs:

1. Input Layer: The input layer represents the features or attributes of the input data, similar to ELMs.
2. Hidden Layers: MELMs have multiple hidden layers, each consisting of randomly initialized neurons. The number of hidden layers and the number of neurons in each layer can be customized based on the complexity of the problem and the desired architecture.
3. Activation Functions: An activation function is applied to the weighted sum of inputs for each neuron in each hidden layer, introducing nonlinearity. Common activation functions used in MELMs include sigmoid, tanh, or rectified linear unit (ReLU).
4. Output Layer: The output layer produces the final predictions or classifications, using an appropriate activation function.

5. Random Weight Initialization: The weights connecting each layer are randomly generated, similarly to ELMs. Each neuron in the hidden layers has its own set of weights connecting it to the previous layer.
6. Calculate Hidden Layer Outputs: The random weights and the input data are multiplied, and the activation function is applied to obtain the output of each neuron in each hidden layer. The outputs from one hidden layer serve as inputs to the next hidden layer until the final hidden layer is reached.
7. Output Weight Computation: The output weights connecting the last hidden layer to the output layer are computed using a linear regression technique. This involves finding the optimal values of the output weights that minimize the difference between the actual outputs and the desired outputs.
8. Prediction: Once the output weights are obtained, the MELM can make predictions on new data by propagating the input through the network, computing the outputs of each hidden layer, and finally using the output weights to compute the final output.

The introduction of multiple hidden layers in MELMs allows for the creation of a deep neural network architecture. By adding more layers, MELMs can learn hierarchical representations of the input data, capturing more complex relationships and patterns. This makes MELMs well suited for tasks that require higher levels of abstraction and representation learning.

It is worth noting that training MELMs is similar to ELMs, where the random weights connecting the layers are initialized and the output weights are computed using linear regression. However, due to the increased complexity of the network, training deep MELMs might require more data and careful parameter tuning to avoid issues such as vanishing or exploding gradients.

Multilayer extreme learning machines have gained attention in the field of deep learning as they provide a simpler and computationally efficient alternative to traditional backpropagation-based methods. They offer a way to leverage the benefits of deep architectures while maintaining the simplicity and fast learning speed associated with ELMs.

3. The Method

In this section, we present the method we designed related to the adoption of four ELMs, i.e., the ELM, WELM, ELM, and MELM models aimed to biomedical images classification.

The proposed method comprises two main steps: the first, i.e., the training (shown in Figure 1) aims to build a model based on an extreme learning machine while the second step, i.e., the testing (shown in Figure 2) is devoted to evaluating the classification effectiveness of the extreme learning machines model built in the previous step.

The first step in Figure 1 is related to the *Biomedical Image Repository*, as a matter of fact, ELM belongs to the category of machine learning models, which signifies their reliance on data-driven approaches. Consequently, they necessitate a substantial quantity of unbiased and accurately labeled data, particularly when applied to biomedical images.

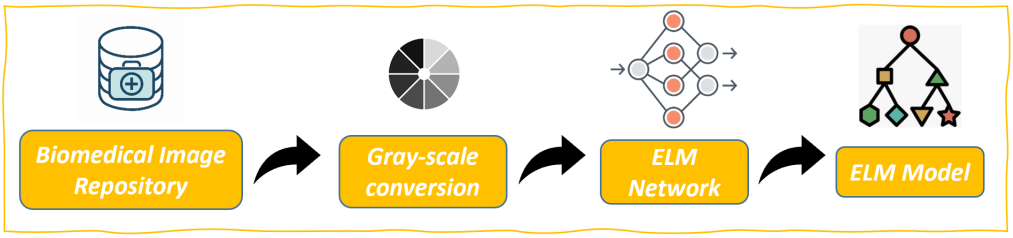

Figure 1. The model training step.

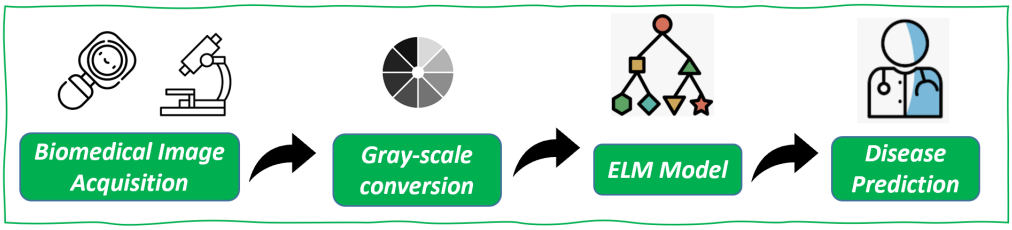

Figure 2. The model testing step.

In the second step, the images, if not already in grayscale, are converted into grayscale format. This conversion has been performed in several previous studies in the biomedical images field that adopted the grayscale conversion [31–34]. Additionally, the conversion helps reduce the time required for model generation.

So, once the biomedical gray-scale images were obtained, we resize the images to a dimension equal to 28 pixels of width of 28 pixels of height and store into a vector the pixel belonging to each grey-scale image: in this way, we obtain, for each image, a vector composed by $28 \times 28 = 784$ pixels.

Once the images have been generated (and stored) into numeric vectors, we can use these data to input the *ELM network*.

In Listing 1 we show the Python code snippet related to the ELM model implementation.

Listing 1. The Python code snippet related to the ELM model implementation.

```
1  import numpy as np
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4  from sklearn.preprocessing import OneHotEncoder
5  from sklearn.preprocessing import MinMaxScaler
6  from sklearn.preprocessing import StandardScaler
7  from scipy.linalg import pinv2
8
9  train = pd.read_csv('dataset_training.csv')
10 test = pd.read_csv('dataset_testing.csv')
11 onehotencoder = OneHotEncoder(categories='auto')
12 scaler = StandardScaler()
13 X_train = scaler.fit_transform(train.values[:,1:])
14 y_train = onehotencoder.fit_transform(train.values[:,:1]).toarray()
15 X_test = scaler.fit_transform(test.values[:,1:])
16 y_test = onehotencoder.fit_transform(test.values[:,:1]).toarray()
17 X_test.shape
18 input_size = X_train.shape[1]
19 hidden_size = 2048
20 input_weights = np.random.normal(size=[input_size,hidden_size])
21 biases = np.random.normal(size=[hidden_size])
22 def relu(x):
23     return np.maximum(x, 0, x)
24 def hidden_nodes(X):
25     G = np.dot(X, input_weights)
26     G = G + biases
27     H = relu(G)
28     return H
29 output_weights = np.dot(pinv2(hidden_nodes(X_train)), y_train)
30 def predict(X):
31     out = hidden_nodes(X)
32     out = np.dot(out, output_weights)
33     return out
34 prediction = predict(X_test)
```

In rows from 1 to 7, we load all the necessary Python libraries, i.e., pandas, numpy, sklearn, and scipy. Rows 9 and 10 are aimed to load the training and testing datasets aiming to, respectively, train and test the ELM model. In fact, the training dataset is considered to build the model (i.e., to perform the training step of the proposed method), while the testing dataset is exploited to accomplish the testing step, aiming to evaluate the effectiveness of the ELM model: to this aim, we need to split the dataset previously obtained into two different sets of images: training and testing.

We use a "OneHotEncoder" (in row 11) to transform our targets into one hot encoding and 'MinMaxScaler' (in row 12) to normalize the features within the range of (0, 1)

In row 18, we define the size of the input layer as referring to the number of input features of the dataset.

Row 19 is related to the initialization of the number of hidden neurons to 1000.

Next, we need to initialize input weights and biases randomly drawn from a Gaussian distribution, as shown in rows 20 and 21.

The relu function, in rows 22 and 23, implements the hidden layer activation function.

The hidden_nodes function, in rows 24–28, is related to the implementation of hidden nodes.

The ELM network, the code of which is shown in Snippet 1, aims to generate the ELM model.

In Figure 2, the testing step is depicted, aiming to evaluate whether the ELM model built in the previous phase is effective in biomedical image classification.

To assess the effectiveness of the ELM model, a separate set of images that were not used in the training process is employed. For example, a biomedical image obtained from a dermatoscope or a microscope is acquired. The acquired image is then converted into grayscale, and its dimensions are resized to 28×28 for further analysis and then we input this image into the ELM model: the model will output the label related to the image by performing the classification.

4. Experimental Analysis

To assess the performance of the proposed models, four distinct metrics are calculated: precision, recall, F-Measure, and accuracy. These metrics provide quantitative measures that evaluate the different aspects of the models' performance.

Precision, as defined in this context, refers to the proportion of individuals who test negative among the total number of individuals who do not have the disease. It quantifies the accuracy of correctly identifying individuals who are truly disease-free among those who are predicted as negative:

$$Precision = \frac{tn}{tn + fp}$$

where tn indicates the number of true negatives and fp is related to the number of false positives.

Recall, in the context of classification, represents the proportion of individuals who test positive among the total number of individuals who actually have the disease. It measures the ability of the classification model to correctly identify individuals who are truly positive for the disease among the entire population of individuals who have the disease:

$$Recall = \frac{tp}{tp + fn}$$

where tp indicates the number of true positives and fn indicates the number of false negatives.

The F-Measure is a metric that captures the weighted average between the precision and recall measures. It provides a balanced evaluation of the model's performance by considering both the model's ability to correctly identify negative instances and its ability to correctly identify positive instances:

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Accuracy, in its definition, encompasses both types of observational errors, namely random and systematic errors. Consequently, achieving high accuracy necessitates both high precision and high recall. In other words, a highly accurate model should exhibit both a low rate of false positives (high precision) and a low rate of false negatives (high recall):

$$Accuracy = \frac{tn + tp}{tn + tp + fn + fp}$$

We compute the metrics with the aim of evaluating the ELM for the four case studies we designed. Moreover, as stated in the introduction, we compare the ELM models, i.e., ELM, WELM, ELM, and MELM with the AlexNet one. AlexNet is a CNN architecture composed of eight layers: the first five were convolutional layers, whilst some of them were followed by max-pooling layers, and the last three were fully connected layers. This used the non-saturating ReLU activation function, which showed an improved training performance over tanh and sigmoid [25]. The AlexNet model was trained for one epoch, with an image size equal to 224 × 224 with three RGB channels. Moreover, with the aim to perform a better comparison with respect to the four ELM architectures, we perform experiments with four widespread supervised machine learning algorithms: J48, Random Forest, REPTree, Bayes, and Naive Bayes [35].

To perform the experiment, we considered a machine with an i7 8th Generation Intel CPU and 16 GB RAM memory, equipped with Microsoft Windows 10 and running the Windows Subsystem for Linux: the model training lasted 4 h, 14 min, and 48.87 s.

The ELM and the AlexNet models were implemented with the Python programming language (version 3.6.9), particularly for the AlexNet model implementation, for which we exploited the Tensorflow 2.4.4 library [36].

4.1. The Datasets

In order to evaluate the ELM adoption in biomedical image classification tasks, we resort to two different datasets belonging to the MedMNIST https://medmnist.com/ [37] (accessed on 20 July 2023) repository, a collection of standardized biomedical images from different organs and acquired with different instruments.

All images are pre-processed into 28 × 28 (i.e., 2D) with the corresponding classification labels. In particular, we exploit biomedical acquires with two different biomedical equipment: dermatoscope and blood cell microscope, with the aim of showing that ELM can be generally exploited for (multiclass and binary) biomedical image classification tasks.

For each classification, a k-fold cross-validation process is exploited. K-fold cross-validation is a common technique used in machine learning to assess the performance and generalization ability of a model. It involves partitioning the dataset into K subsets or folds of approximately equal size, where K is a predefined integer value. The training and evaluation process is then repeated K times, with each fold serving as the validation set once, while the remaining K-1 folds are used for training. In particular, we set k = 10.

The first dataset we take into account is the DermaMNIST one: it is based on the HAM10000 [38,39], a large collection of multi-source dermatoscopic images of common pigmented skin lesions. The DermaMNIST dataset comprises 10,015 dermatoscopic images, out of which 8010 images are designated for training purposes, while the remaining 2005 images are allocated for testing. The dataset includes images classified into seven distinct diseases. To divide the dataset into training and testing sets, an 80:20 ratio is employed. The original source images, which have dimensions of 3 × 600 × 450, are resized to 3 × 28 × 28. This resizing results in each image having dimensions of 28 × 28 with three RGB channels.

In particular, the DermaMNIST dataset is composed by the following labels (exploited in the multiclass classification) to discriminate between the different diseases affecting the Derma:

- The 0 label is related to actinic keratoses and intraepithelial carcinoma;
- The 1 label is related to basal cell carcinoma;
- The 2 label is related to benign keratosis-like lesions;
- The 3 label is related to dermatofibroma;
- The 4 label is related to melanoma;
- The 5 label is related to melanocytic nevi;

- The 6 label is related to vascular lesions.

With regard to the DermaMNIST binary classification, we consider the two following labels:

- The 0 label is related to following diseases, including actinic keratoses and intraepithelial carcinoma, basal cell carcinoma, benign keratosis-like lesions, dermatofibroma, melanocytic nevi, and 'vascular lesions';
- The 1 label is related to melanoma (this case study is related to melanoma disease detection).

The second dataset we consider is the BloodMNIST one, which belongs to the MedMINST collection as the DermaMinst dataset. The BloodMNIST dataset [40] is derived from a collection of individual normal blood cells obtained from individuals who do not have any infections, hematologic or oncologic diseases, and are not undergoing any pharmacologic treatment at the time of blood collection. The dataset consists of a total of 17,092 images, among which 13,671 images are used for training the model, while the remaining 3412 images are reserved for model testing. The dataset is categorized into eight different classes. To divide the source dataset into training and testing sets, an 80:20 ratio is employed. The source images, initially with a resolution of 3 × 360 × 363 pixels, are first center-cropped to 3 × 200 × 200 pixels. Subsequently, they are resized to 3 × 28 × 28 pixels. This resizing results in images with a width and height of 28 pixels and each pixel is represented in three channels (RGB format).

The BloodMNIST dataset is composed of the following labels (exploited in the multiclass classification) to perform the blood cell detection task:

- With the label 0, images are associated with basophil;
- With the label 1, images are associated with eosinophil;
- With the label 2, images are associated with erythroblast;
- With the label 3, images are associated with immature granulocytes, i.e., myelocytes, metamyelocytes, and promyelocytes;
- With the label 4, images are associated with lymphocyte;
- With the label 5, images are associated with monocyte;
- With the label 6, images are associated with neutrophil;
- With the label 7, images are associated with platelet.

With regard to the BloodMNIST dataset binary classification, we consider the following two labels:

- With the label 0, we consider the basophil, eosinophil, erythroblast, immature granulocytes, lymphocyte, monocyte, and neutrophil cells;
- With the label 1, we consider the platelets, i.e., the idea is to predict whether a blood microscopic image is containing a platelet or another type of blood cell.

4.2. Binary Classification Results

In this section, we report the results we obtained in the binary classification task by exploiting both the DermaMINST and the BloodMNIST datasets.

In particular, the first binary model is related to the melanoma detection: we consider $T_{detection}$ as a set of labels $\{(M_{detection}, l_{detection})\}$, where each $M_{detection}$ is the label that is associated with a $l_{detection} \in \{0, 1\}$, where 1 denotes melanoma, while 0 denotes other pathologies (i.e., actinic keratoses, intraepithelial carcinoma, basal cell carcinoma, benign keratosis-like lesions, dermatofibroma, melanocytic nevi, and vascular lesions).

We build a numeric vector of features $F \in R_y$, where y represents the feature number exploited in the learning phase ($y = 784$, as a matter of fact, for each 28 × 28 gray-scale image under analysis, we consider each pixel as a feature).

Table 1 shows the classification results obtained from binary classification with the DermaMINST dataset with the ELM, WELM, ELM, and MELM models. For each ELM model, we consider four different hidden neurons (i.e., column hn in Table 1): 50, 100, 500, and 1000. We also report the time required for the classification (i.e., column Time in Table 1).

As shown in Table 1, all the ELM models are able to reach interesting performances, very similar to each other. For instance, the ELM model with 1000 hidden neurons obtains an accuracy equal to 0.884, while with 50 hidden neurons, the accuracy is equal to 0.888. Also, the W-ELM, R-ELM, and ML-ELM models obtain an accuracy ranging from 0.887 to 0.891, by confirming the effectiveness of ELM in the task of the binary classification of the DermaMINST dataset.

Table 1. DermaMINST binary ELM classification results.

Alg	hn	Precision	Recall	F-Measure	Accuracy	Time
ELM	50	0.846	0.888	0.838	0.888	0:00:1.55
	100	0.831	0.887	0.839	0.887	0:00:1.56
	500	0.846	0.886	0.850	0.886	0:00:2.44
	1000	0.850	0.884	0.857	0.884	0:00:4.56
W-ELM	50	0.827	0.887	0.837	0.887	0:00:1.10
	100	0.879	0.891	0.844	0.891	0:00:1.29
	500	0.849	0.884	0.856	0.884	0:00:4.47
	1000	0.841	0.880	0.852	0.880	0:00:4.48
R-ELM	50	0.901	0.889	0.838	0.889	0:00:0.88
	100	0.875	0.890	0.843	0.890	0:00:0.92
	500	0.859	0.890	0.856	0.890	0:00:1.07
	1000	0.841	0.879	0.852	0.879	0:00:1.46
ML-ELM	50	0.874	0.889	0.839	0.889	0:00:0.95
	100	0.840	0.887	0.842	0.887	0:00:1.01
	500	0.860	0.891	0.856	0.891	0:00:1.59
	1000	0.860	0.890	0.863	0.890	0:00:3.07

The confusion matrix is shown in Figure 3 for DermaMinst binary classification with the ELM model.

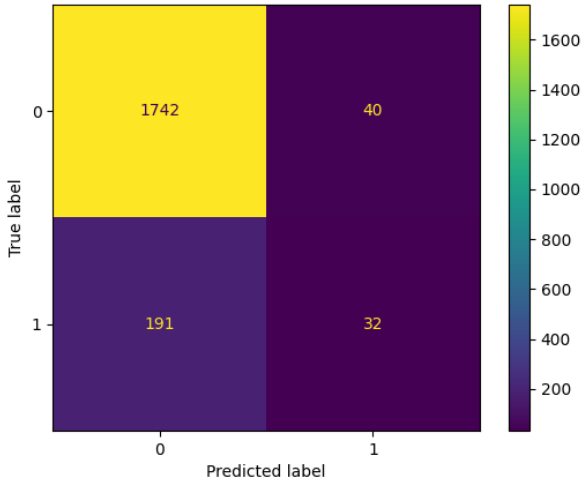

Figure 3. The DermaMINST binary classification confusion matrix.

From the confusion matrix, it emerges that the ELM model is able to correctly classify 32 images related to melanoma and 1742 images related to other pathologies affecting the Derma.

In Table 2, we show the results obtained with the machine learning algorithms we exploited for comparison.

As shown from Table 2, the ELM models are able to obtain better performances with respect to the AlexNet one: as a matter of fact, the ELM model reaches an accuracy equal to 0.872, while the AlexNet model obtains an accuracy of 0.343. Moreover, the ELM model requires 33 s for training, while the AlexNet one more than 13 min. This also happens

when we consider the J48, the Random Forest, the REPTree, the Bayes, and the Naive Bayes classification algorithm: in fact, an accuracy ranging from 0.828 to 0.864 is obtained with these models, thus showing that ELM models obtain better accuracy in this case study.

Table 2. DermaMINST binary machine learning classification results.

Model	Precision	Recall	F-Measure	Accuracy	Time
J48	0.829	0.826	0.827	0.828	0:00:51.84
Random Forest	0.874	0.890	0.841	0.851	0:00:21.66
REPTree	0.824	0.886	0.839	0.864	0:00:9.42
Bayes	0.837	0.682	0.738	0.778	0:00:5.46
Naive Bayes	0.835	0.692	0.744	0.789	0:00:1.52
AlexNet	0.343	0.343	0.320	0.343	0:13:23.77

In the next, we describe the classification process and results we obtained with the binary classification with the BloodMINST dataset. This second case study is related to platelets' detection: in this case, we consider $T_{detection}$ as a set of labels $\{(M_{detection}, l_{detection})\}$ where each $M_{detection}$ is the label that is associated with a $l_{detection} \in \{0, 1\}$, where 1 is related to platelets, while 0 with the other blood cells (i.e., basophil, eosinophil, erythroblast, immature granulocytes, lymphocyte, monocyte, and neutrophil).

In Table 3, we show the results of the binary classification performed with the BloodMINST dataset.

Table 3. BloodMINST binary ELM classification results.

Alg	hn	Precision	Recall	F-Measure	Accuracy	Time
ELM	50	0.846	0.888	0.838	0.888	0:00:1.55
	100	0.980	0.980	0.980	0.980	0:00:1.48
	500	0.996	0.996	0.996	0.996	0:00:3.88
	1000	0.995	0.995	0.995	0.995	0:00:9.37
W-ELM	50	0.975	0.975	0.975	0.975	0:00:1.89
	100	0.991	0.991	0.991	0.991	0:00:3.76
	500	0.995	0.995	0.995	0.995	0:00:4.96
	1000	0.995	0.995	0.995	0.995	0:00:9.39
R-ELM	50	0.979	0.980	0.979	0.980	0:00:2.20
	100	0.987	0.987	0.987	0.987	0:00:2.09
	500	0.997	0.997	0.997	0.997	0:00:2.41
	1000	0.997	0.997	0.997	0.997	0:00:3.40
ML-ELM	50	0.982	0.982	0.982	0.982	0:00:1.54
	100	0.987	0.987	0.987	0.987	0:00:1.52
	500	0.995	0.995	0.995	0.995	0:00:2.37
	1000	0.995	0.995	0.995	0.995	0:00:4.37

From Table 3, it can be seen that ELM, W-ELM, R-ELM, and ML-ELM models are able to reach good performances, as a matter of fact, the accuracy is ranging from 0.888 to 0.997: in this case, the best result (i.e., 0.997) is obtained from the R-ELM model, when 500 and 1000 hidden neurons are used. Also, the ML-ELM model obtained good performance, with an accuracy equal to 0.995 when 500 and 1000 hidden neurons are considered.

The confusion matrix related to the BloodMINST binary classification with the ELM model with 1000 hidden neurons is shown in Figure 4.

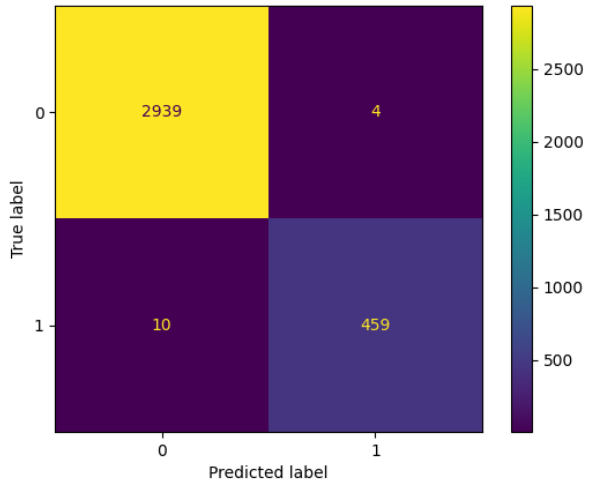

Figure 4. The BloodMINST binary classification confusion matrix.

As shown in the confusion matrix in Figure 4, 459 platelets are correctly classified together with 2939 blood microscopic images related to other cells.

Table 4 shows the comparison results related to the BloodMINST binary classification.

Table 4. BloodMINST binary machine learning classification results.

Model	Precision	Recall	F-Measure	Accuracy	Time
J48	0.990	0.990	0.990	0.990	0:00:32.89
Random Forest	0.997	0.997	0.997	0.997	0:00:23.12
REPTree	0.989	0.989	0.989	0.989	0:00:9.08
Bayes	0.980	0.977	0.978	0.978	0:00:21.08
Naive Bayes	0.981	0.979	0.980	0.980	0:00:2.78
AlexNet	0.991	0.991	0.990	0.991	0:18:45.48

As shown in the results in Table 4, good performances are also reached from the AlexNet model: in particular, an accuracy equal to 0.991 is reached with the AlexNet model, confirming that the ELM model is able to obtain (in this case, slightly) better performances than the AlexNet one. From the time point of view, the ELM model employs 28 s to model the training and testing, while the AlexNet one requires 18 min and 45 min. Relating to the J48, Random Forest, REPTree, Bayes, and BayesNet performances, in this case, are also able to obtain good performances, with an accuracy ranging from 0.980 to 0.997 (with the Random Forest model); the same is obtained when the R-ELM model is considered, but when the R-ELM requires approximately 3 s for the training and testing, the Random Forest requires 23 s, thus confirming the effectiveness of ELM models.

4.3. Multiclass Classification Results

In the following, we describe the classification process and results we obtained with the multiclass classification, respectively, with the DermaMINST and the BloodMINST dataset.

This third case study is related to the classification of different pathologies afflicting the Derma detection with the DermaMINST dataset: to this aim, we consider $T_{detection}$ as a set of labels $\{(M_{detection}, l_{detection})\}$, where each $M_{detection}$ is the label that is associated with a $l_{detection} \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, where 0 is related to actinic keratoses and intraepithelial carcinoma, 1 denotes 'basal cell carcinoma, 2 denotes benign keratosis-like lesions, 3 denotes dermatofibroma, 4 denotes melanoma, 5 denotes ' melanocytic nevi, and 6 denotes vascular lesions.

Table 5 shows the results obtained in the DermaMINST multiclass classification with the ELM, W-ELM, R-ELM, and ML-ELM models.

Table 5. DermaMINST multiclass ELM classification results.

Alg	hn	Precision	Recall	F-Measure	Accuracy	Time
ELM	50	0.520	0.670	0.554	0.670	0:00:1.78
	100	0.562	0.677	0.578	0.677	0:00:1.08
	500	0.590	0.683	0.609	0.683	0:00:1.72
	1000	0.588	0.674	0.614	0.674	0:00:3.07
W-ELM	50	0.526	0.670	0.554	0.670	0:00:1.10
	100	0.597	0.677	0.586	0.677	0:00:1.32
	500	0.593	0.681	0.609	0.681	0:00:2.44
	1000	0.593	0.678	0.620	0.678	0:00:4.48
R-ELM	50	0.532	0.668	0.554	0.668	0:00:0.91
	100	0.535	0.670	0.567	0.670	0:00:0.94
	500	0.570	0.674	0.596	0.674	0:00:1.19
	1000	0.580	0.665	0.604	0.665	0:00:1.40
ML-ELM	50	0.542	0.668	0.553	0.668	0:00:1.49
	100	0.608	0.679	0.580	0.679	0:00:1.51
	500	0.622	0.687	0.611	0.687	0:00:2.43
	1000	0.605	0.683	0.621	0.683	0:00:4.43

As shown from the results shown in Table 5, the accuracy ranges from 0.670 to 0.683. This happens for all the ELM models considered.

Figure 5 shows the confusion matrix for the third case study.

The DermaMINST multiclass classification confusion matrix for the ELM model (trained with 1000 hidden neurons) is shown in Figure 5.

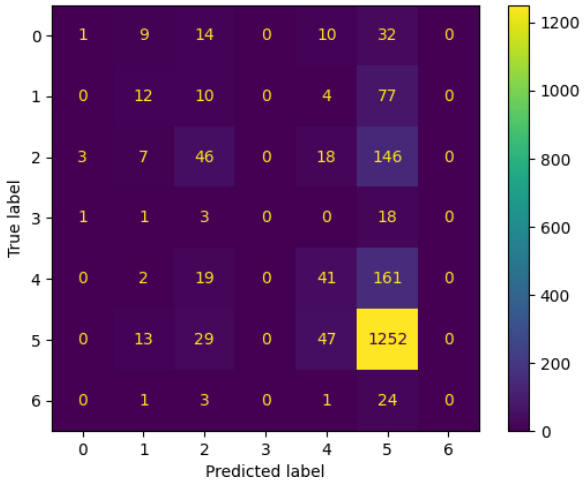

Figure 5. The DermaMINST multiclass classification confusion matrix.

From the confusion matrix in Figure 5, we can observe that a large part of the derma images is correctly classified in the right class, thus confirming the effectiveness of ELM.

Table 6 shows the results we obtained in comparison with machine learning in the DermaMINST multiclass classification.

From the classification results in Table 6, it can be seen that the ELM models are able to obtain (slightly) better performances if compared to the AlexNet (able to reach an accuracy equal to 0.668). The ELM model requires 23 s for the training and testing of the model, while the AlexNet needs 26 min for the same task. Moreover, with regard to J48, Random Forest, REPTree, Bayes, and Naive Bayes models, an accuracy ranging from 0.31 to 568 is

obtained, also confirming for these models that ELM networks are able to obtain (slightly) better performances for the DermaMINST multiclass classification.

Table 6. DermaMINST multiclass machine learning classification results.

Model	Precision	Recall	F-Measure	Accuracy	Time (s)
J48	0.567	0.569	0.568	0.568	0:00:26.66
Random Forest	0.512	0.684	0.515	0.514	0:00:15.78
REPTree	0.559	0.663	0.589	0.568	0:00:6.32
Bayes	0.608	0.225	0.244	0.312	0:00:2.81
Naive Bayes	0.611	0.220	0.246	0.327	0:00:2.11
AlexNet	0.655	0.317	0.425	0.668	0:26:00.99

The last case study we present is related to multiclass classification with the BloodMINST dataset. The idea of this fourth case study is the identification of different blood cells: to this aim, we consider $T_{detection}$ as a set of labels $\{(M_{detection}, l_{detection})\}$, where each $M_{detection}$ is the label that is associated with a $l_{detection} \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, where the label 0 denotes basophil, the label 1 denotes eosinophil, the label 2 denotes erythroblast, the label 3 denotes immature granulocytes, the label 4 denotes lymphocyte, the label 5 denotes monocyte, the label 6 denotes neutrophil, and the label 7 denotes platelet.

The results related to the fourth case study, i.e., the BloodMINST multiclass classification with ELM, W-ELM, R-ELM, and ML-ELM models, are shown in Table 7.

Table 7. BloodMINST multiclass ELM classification results.

Alg	hn	Precision	Recall	F-Measure	Accuracy	Time
ELM	50	0.490	0.502	0.463	0.502	0:00:1.45
	100	0.572	0.572	0.549	0.572	0:00:1.96
	500	0.660	0.661	0.652	0.661	0:00:2.50
	1000	0.696	0.697	0.690	0.697	0:00:4.74
W-ELM	50	0.503	0.513	0.487	0.513	0:00:2.18
	100	0.575	0.573	0.549	0.573	0:00:2.27
	500	0.662	0.662	0.651	0.662	0:00:4.72
	1000	0.683	0.684	0.675	0.684	0:00:9.20
R-ELM	50	0.507	0.523	0.488	0.523	0:00:1.40
	100	0.549	0.556	0.532	0.556	0:00:1.43
	500	0.669	0.665	0.657	0.665	0:00:1.66
	1000	0.690	0.693	0.685	0.693	0:00:2.02
ML-ELM	50	0.486	0.502	0.464	0.502	0:00:2.40
	100	0.567	0.568	0.544	0.568	0:00:2.34
	500	0.675	0.675	0.678	0.675	0:00:3.69
	1000	0.684	0.686	0.679	0.686	0:00:6.56

As shown from Table 7, in this last case study, it can be seen that the accuracy obtained ranges from 0.502 to 0.697 for the ELM, W-ELM, R-ELM, and ML-ELM models. In particular, quite satisfactory accuracy is obtained from the ELM model, trained with 1000 hidden neurons, equal to 0.697.

Figure 6 shows the confusion matrix for the third case study when the ELM model with 1000 hidden neurons is considered.

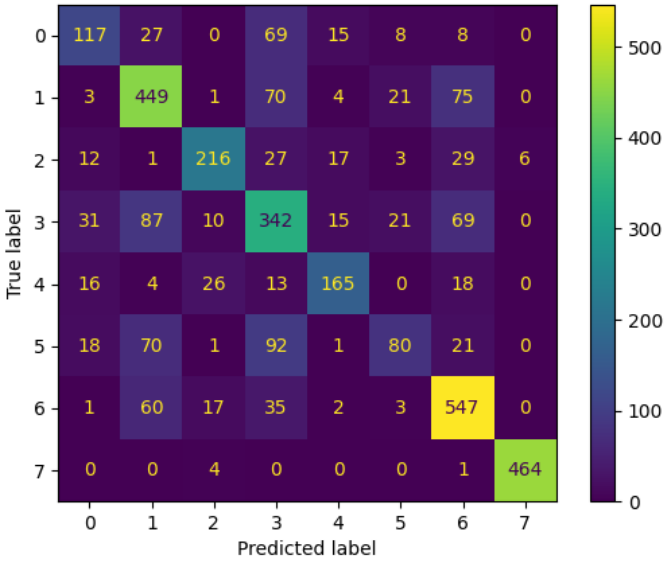

Figure 6. The BloodMINST multiclass classification confusion matrix.

As shown from the BloodMINST multiclass classification, we can note that a large number of instances are correctly classified in the right cell class, confirming the effectiveness of ELM in blood cell identification.

Table 8 shows the comparison with respect to machine learning for the BloodMINST multiclass classification.

Table 8. BloodMINST multiclass machine learning classification results.

Model	Precision	Recall	F-Measure	Accuracy	Time (s)
J48	0.625	0.624	0.624	0.623	0:00:43.11
Random Forest	0.806	0.801	0.794	0.796	0:00:26.86.78
REPTree	0.643	0.646	0.642	0.642	0:00:15.67
Bayes	0.769	0.753	0.754	0.755	0:00:14.24
Naive Bayes	0.690	0.677	0.671	0.673	0:00:2.2
AlexNet	0.984	0.073	0.134	0.269	0:26:00.99

From Table 8, it can be seen that the ELM model obtains a better accuracy than the AlexNet model, which is able to reach an accuracy of 0.269, with an amount of time equal to 39 s for training and testing with the ELM model and equal to 26 s for AlexNet. Relating to the remaining models (i.e., J48, Random Forest, REPTree, Bayes, and Naive Bayes), an accuracy ranging from 0.623 to 0.755 is reached; therefore, we note that, in this last case study, in some cases, slightly higher accuracy values are reached compared to the ELM models: in any case, we note that the training time is always less for the ELM models.

As seen from the experimental results in the four case studies, the ELM model is able to obtain even better accuracy than the AlexNet network in all the case studies, which also applies to most of the machine learning models considered in this experiment, allowing significantly reduced times for the model training and testing phases.

5. Conclusions and Future Work

ELM represents an emerging paradigm compared with classic machine learning models that is able to build predictive models with a performance comparable (but also much higher) to classic machine learning models in a much shorter time. We experiment with the possibility to adopt ELM (in particular, we consider four different models, namely

ELM, W-ELM, R-ELM, and ML-ELM) for biomedical image classification tasks, where researchers consider deep learning (particularly with regard to CNN) to build predictive models. Four different case studies are proposed, the first one exploiting the DermaMINST dataset with a binary classification; the second one using the BloodMINST dataset with a binary classification; the third one considering the DermaMINST dataset with a multiclass classification; and the last one considering the BloodMINST dataset with a multiclass classification. All four case studies, demonstrating that ELM is able to obtain better performances if compared to the AlexNet model, one of the most widespread deep learning models used for biomedical image classification and five different supervised machine learning models, i.e., J48, Random Forest, REPTree, Bayes, and Naive Bayes. In future work, we plan to extend the experiments by considering biomedical images obtained with other acquisition systems (as a matter of fact, we considered images obtained with the dermascope and the blood cell microscope); for instance, optical coherence tomography and computed tomography. Additionally, we will consider a greater ELM architecture by performing an extensive experimental analysis to understand which specific ELM model can be more suitable for a specific biomedical image classification problem.

Author Contributions: Methodology, F.M. (Francesco Mercaldo); Software, F.M. (Francesco Mercaldo) and M.C.; Validation, F.M. (Francesco Mercaldo), L.B., F.M. (Fabio Martinelli) and M.C.; Formal analysis, F.M. (Francesco Mercaldo) and F.M. (Fabio Martinelli); Investigation, A.S.; Data curation, L.B.; Writing—original draft, A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by *MUR - REASONING:* foRmal mEthods for computAtional analySis for diagnOsis and progNosis in imagING - PRIN, National Plan for NRRP Complementary Investments D^3 4 Health: Digital Driven Diagnostics, prognostics and therapeutics for sustainable Health care and *e-DAI* (Digital ecosystem for integrated analysis of heterogeneous health data related to high-impact diseases: innovative model of care and research), Health Operational Plan, FSC 2014-2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in this study.

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

References

1. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
2. Hagan, M.T.; Menhaj, M.B. Training feedforward networks with the Marquardt algorithm. *IEEE Trans. Neural Netw.* **1994**, *5*, 989–993. [CrossRef]
3. Chen, S.; Cowan, C.; Grant, P. Orthogonal Least Squares Learning Algorithm for Radial. *IEEE Trans. Neural Netw.* **1991**, *2*, 303. [CrossRef] [PubMed]
4. Urgen Branke, J. Evolutionary algorithms for neural network design and training. In Proceedings of the 1st Nordic Workshop on Genetic Algorithms and its Applictions, Vaasa, Finland, 9–12 January 1995.
5. Huang, G.; Huang, G.B.; Song, S.; You, K. Trends in extreme learning machines: A review. *Neural Netw.* **2015**, *61*, 32–48. [CrossRef] [PubMed]
6. Huang, G.B.; Chen, L. Convex incremental extreme learning machine. *Neurocomputing* **2007**, *70*, 3056–3062. [CrossRef]
7. Huang, G.B.; Chen, L.; Siew, C.K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Netw.* **2006**, *17*, 879–892. [CrossRef] [PubMed]
8. Liu, X.; Lin, S.; Fang, J.; Xu, Z. Is extreme learning machine feasible? A theoretical assessment (Part I). *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *26*, 7–20. [CrossRef]
9. Lin, S.; Liu, X.; Fang, J.; Xu, Z. Is extreme learning machine feasible? A theoretical assessment (Part II). *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *26*, 21–34. [CrossRef]
10. Huang, G.B.; Zhou, H.; Ding, X.; Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. Part B* **2011**, *42*, 513–529. [CrossRef]
11. Cortes, C.; Vapnik, V. Support vector machine. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
12. Suykens, J.A.; Vandewalle, J. Least squares support vector machine classifiers. *Neural Process. Lett.* **1999**, *9*, 293–300. [CrossRef]

13. Fernández-Delgado, M.; Cernadas, E.; Barro, S.; Ribeiro, J.; Neves, J. Direct Kernel Perceptron (DKP): Ultra-fast kernel ELM-based classification with non-iterative closed-form weight calculation. *Neural Netw.* **2014**, *50*, 60–71. [CrossRef] [PubMed]
14. Huang, G.; Song, S.; Gupta, J.N.; Wu, C. Semi-supervised and unsupervised extreme learning machines. *IEEE Trans. Cybern.* **2014**, *44*, 2405–2417. [CrossRef] [PubMed]
15. Huang, G.B. An insight into extreme learning machines: Random neurons, random features and kernels. *Cogn. Comput.* **2014**, *6*, 376–390. [CrossRef]
16. Huang, G.; Song, S.; Wu, C. Orthogonal least squares algorithm for training cascade neural networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2012**, *59*, 2629–2637. [CrossRef]
17. Reddy, M.B.S.; Rana, P. Biomedical image classification using deep convolutional neural networks–overview. In *Proceedings of the IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK , 2021; Volume 1022, p. 012020.
18. Deepa, S.; Devi, B.A. A survey on artificial intelligence approaches for medical image classification. *Indian J. Sci. Technol.* **2011**, *4*, 1583–1595. [CrossRef]
19. Ali, M.; Sarwar, A.; Sharma, V.; Suri, J. Artificial neural network based screening of cervical cancer using a hierarchical modular neural network architecture (HMNNA) and novel benchmark uterine cervix cancer database. *Neural Comput. Appl.* **2019**, *31*, 2979–2993. [CrossRef]
20. Urushibara, A.; Saida, T.; Mori, K.; Ishiguro, T.; Sakai, M.; Masuoka, S.; Satoh, T.; Masumoto, T. Diagnosing uterine cervical cancer on a single T2-weighted image: Comparison between deep learning versus radiologists. *Eur. J. Radiol.* **2021**, *135*, 109471. [CrossRef]
21. Mohsen, H.; El-Dahshan, E.S.A.; El-Horbaty, E.S.M.; Salem, A.B.M. Classification using deep learning neural networks for brain tumors. *Future Comput. Inform. J.* **2018**, *3*, 68–71. [CrossRef]
22. Afshar, P.; Mohammadi, A.; Plataniotis, K.N. Brain tumor type classification via capsule networks. In Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP), Athens, Greece, 7–10 October 2018, pp. 3129–3133.
23. Zia, R.; Akhtar, P.; Aziz, A. A new rectangular window based image cropping method for generalization of brain neoplasm classification systems. *Int. J. Imaging Syst. Technol.* **2018**, *28*, 153–162. [CrossRef]
24. Jain, M.; Andreopoulos, W.; Stamp, M. Convolutional neural networks and extreme learning machines for malware classification. *J. Comput. Virol. Hacking Tech.* **2020**, *16*, 229–244. [CrossRef]
25. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
26. Lu, S.; Lu, Z.; Zhang, Y.D. Pathological brain detection based on AlexNet and transfer learning. *J. Comput. Sci.* **2019**, *30*, 41–47. [CrossRef]
27. Dhar, P.; Dutta, S.; Mukherjee, V. Cross-wavelet assisted convolution neural network (AlexNet) approach for phonocardiogram signals classification. *Biomed. Signal Process. Control* **2021**, *63*, 102142. [CrossRef]
28. Kumar M, A.; Chakrapani, A. Classification of ECG signal using FFT based improved Alexnet classifier. *PLoS ONE* **2022**, *17*, e0274225. [CrossRef]
29. Toprak, A. Extreme learning machine (elm)-based classification of benign and malignant cells in breast cancer. *Med. Sci. Monit. Int. Med. J. Exp. Clin. Res.* **2018**, *24*, 6537. [CrossRef]
30. Murugan, R.; Goel, T. E-DiCoNet: Extreme learning machine based classifier for diagnosis of COVID-19 using deep convolutional network. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 8887–8898. [CrossRef] [PubMed]
31. Pavlov, S.; Vassilenko, V.; Saldan, I.; Vovkotrub, D.; Poplavskaya, A.; Kuzin, O. Methods of processing biomedical image of retinal macular region of the eye. In Proceedings of the Reflection, Scattering, and Diffraction from Surfaces V, Online, 24 August–4 September 2016; Volume 9961, pp. 253–260.
32. Kapoor, L.; Thakur, S. A survey on brain tumor detection using image processing techniques. In Proceedings of the 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, Noida, India, 12–13 January 2017; pp. 582–585.
33. Sternberg, S.R. Biomedical image processing. *Computer* **1983**, *16*, 22–34. [CrossRef]
34. Kaucha, D.P.; Prasad, P.; Alsadoon, A.; Elchouemi, A.; Sreedharan, S. Early detection of lung cancer using SVM classifier in biomedical image processing. In Proceedings of the 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, 21–22 September 2017; pp. 3143–3148.
35. Learning, M. *Tom Mitchell*; McGraw Hill: New York, NY, USA, 1997.
36. Singh, P.; Manure, A.; Singh, P.; Manure, A. Introduction to tensorflow 2.0. In *Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python* ; 2020; pp. 1–24. Available online: https://books.google.com.hk/books?hl=en&lr=&id=3_rEDwAAQBAJ&oi=fnd&pg=PR5&dq=Learn+TensorFlow+2.0:+Implement+Machine+Learning+and+Deep+++Learning+Models+with+Python&ots=iac1aMg5mX&sig=C1uvgYzaOVuCQWH2DzGnMIHhGK8&redir_esc=y (accessed on 20 July 2023).
37. Yang, J.; Shi, R.; Ni, B. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In Proceedings of the 2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI), Nice, France, 13–16 April 2021; pp. 191–195.
38. Tschandl, P.; Rosendahl, C.; Kittler, H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Sci. Data* **2018**, *5*, 180161. [CrossRef] [PubMed]

39. Codella, N.; Rotemberg, V.; Tschandl, P.; Celebi, M.E.; Dusza, S.; Gutman, D.; Helba, B.; Kalloo, A.; Liopyris, K.; Marchetti, M.; et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv* **2019**, arXiv:1902.03368.
40. Acevedo, A.; Merino, A.; Alférez, S.; Molina, Á.; Boldú, L.; Rodellar, J. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. *Data Brief* **2020**, *30*, 105474. [CrossRef] [PubMed]