Neural Computing and Applications (2021) 33:15121–15144
https://doi.org/10.1007/s00521-021-06402-y

REVIEW ARTICLE

Extreme learning machine versus classical feedforward network

Comparison from the usability perspective

Urszula Markowska-Kaczmar[1] · Michał Kosturek[1]

Received: 25 March 2021 / Accepted: 29 July 2021 / Published online: 30 August 2021

Abstract
Our research is devoted to answering whether randomisation-based learning can be fully competitive with the classical feedforward neural networks trained using backpropagation algorithm for classification and regression tasks. We chose extreme learning as an example of randomisation-based networks. The models were evaluated in reference to training time and achieved efficiency. We conducted an extensive comparison of these two methods for various tasks in two scenarios: • using comparable network capacity and • using network architectures tuned for each model. The comparison was conducted on multiple datasets from public repositories and some artificial datasets created for this research. Overall, the experiments covered more than 50 datasets. Suitable statistical tests supported the results. They confirm that for relatively small datasets, extreme learning machines (ELM) are better than networks trained by the backpropagation algorithm. But for demanding image datasets, like ImageNet, ELM is not competitive to modern networks trained by backpropagation; therefore, in order to properly address current practical needs in pattern recognition entirely, ELM needs further development. Based on our experience, we postulate to develop smart algorithms for the inverse matrix calculation, so that determining weights for challenging datasets becomes feasible and memory efficient. There is a need to create specific mechanisms to avoid keeping the whole dataset in memory to compute weights. These are the most problematic elements in ELM processing, establishing the main obstacle in the widespread ELM application.

Keywords Randomisation-based learning · Extreme learning · Backpropagation algorithm · Neural networks

1 Introduction

Artificial neural networks are among the fastest-growing areas of artificial intelligence. Existing deep models can solve tasks such as visual object recognition or automatic text translation, often achieving human-level performance. Frequently, these fantastic results need weeks of training and require substantial computational power. High training times are a consequence of the most commonly used training strategy—numerical optimisation using iterative methods and the backpropagation algorithm. Its purpose is to determine the error gradient for each parameter of the network. We will refer to them as classical or standard networks. High hardware requirements and long training times are frequent obstacles in applying neural networks in practical use. These difficulties explain our interest in searching for an alternative solution. In this work, we examine on randomisation-based neural networks, which offer significantly lower training times and claim to perform on par with classical networks. Specifically, we focus on extreme learning machines (ELM), a special case of random vector functional-link (RVFL) networks.

1.1 Motivation

Motivation for this research stems from our search for models useful to solve real problems in classification and regression tasks which are well suited to be applied in

✉ Urszula Markowska-Kaczmar
urszula.markowska-kaczmar@pwr.edu.pl

Michał Kosturek
kosturek.michal@gmail.com

[1] Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

practice. We limited our interests to processing inputs in the form of vectors and raw images. We do not consider sequential data.

Random vector functional-link networks were proposed in [32], and their characteristics were discussed in [33]. Since then, they were developed and evaluated [11, 47, 50]. RVFL is a feedforward network with a single hidden layer and direct links between input and output layers which bypass the hidden layer. The most important characteristic of RVFL is that weights assigned to the hidden layer are set randomly and are not optimised. Output layer weights are the only trainable parameters.

Authors of [13] proposed extreme learning machines. ELM follows the same principle as RVFL. In fact, they are a special case of RVFL, where the direct input-output links are disabled [5]. Therefore ELM is architecturally identical to commonly used classical feedforward networks with a single hidden layer. The only difference between them is the training method. Authors of ELM claim that time of assigning weights for ELM is significantly lower than when using standard network training methods and with comparable or even better performance. Extreme learning machines were used as a baseline for many further models, such as stacked ELM [51], on-line sequential ELM [14], LRF-ELM designed for image classification [17], and biased dropout ELM [25]. ELM can also be considered as a non-recurrent equivalent of reservoir computing (RC) models [40]. Because random units construct features (to deal with nonlinearly separable tasks) without learning, this model is a natural candidate to overcome classical neural networks' time-consuming training. This allows to find all remaining weights in a closed form, which means that no iterative numerical optimisation is required.

The use of random parameters in randomisation-based networks may raise concerns about deteriorating the performance quality. However, authors of [16] prove that a sufficiently high number of random hidden features allow them to learn effectively. This statement sounds very promising, but our literature survey does not result in an in-depth comparison of backpropagation and randomisation-based learning—both ELM and RVFL. There are, however, detailed comparisons between RVFL and ELM [9, 43, 50], where these models are compared using multiple datasets for both classification and regression. These comparisons show that using direct links between input and output layers does improve the performance in classification and does not have a significant effect on the quality of regression. Nonetheless, these findings do not show whether randomisation-based learning offers a true alternative for commonly used backpropagation-based learning algorithms.

In the original paper on extreme learning [13], there is a simple comparison between ELMs and networks trained with backpropagation. The comparison covers two datasets—*Diabetes* for classification and *California Housing* for regression, both from the UCI repository [8]. The study ensures identical architectures for both models so that the only properties compared were the training algorithms. A similar comparison was held on the *Forest Type* dataset, but in this case, ELM contained twice the number of hidden neurons the regular network had. Thus, this comparison lacks objectivity. Unfortunately, the authors do not specify hyperparameter set-up, e.g. stopping criterion for backpropagation-based method. This approach causes that the results are non-reproducible. Further works on extreme learning include more comprehensive comparisons [16]. They cover more than 10 datasets for regression obtained from University of Porto repository [46] and several classification datasets. Hyperparameter set-up is described in more detail compared to the original paper on extreme learning [13].

Methods improving the basic RVFL and ELM models [25, 47] usually use their immediate predecessors or simple RVFL and ELM as baselines, omitting backpropagation-based networks.

When proposing the LRF-ELM model, authors of [17] conduct a comparison to a regular convolutional network using only a single dataset—NORB. Therefore, conclusions from such experiments are far from being general. Papers on the extensions of extreme learning usually do not present any comparison to other training methods [53]. They are focused on showing the difference between the proposed extension and the basic ELM. Some other works comparing ELMs to regular deep learning models often use outdated or inadequate network architectures. We can cite the paper [21], where authors performed a comparison in the task of image classification and did not use convolutional networks. When publishing that work (the year 2013), CNNs were just emerging; however, nowadays, they are considered the primary choice for image recognition (classification). Despite the dynamic evolution of CNNs, studies are scarcely comparing them to ELMs.

None of the referred works covered analysis of the impact of hyperparameters set-up on the ELM performance. Therefore, it seems worthy to explore some basic hyperparameters (the number of hidden neurons, choice of an activation function, and the value of regularisation coefficient), because they can profoundly influence results. Only [15] mentions that "Since the input weights and hidden biases of ELM are randomly generated instead of being fine-tuned, ELM usually needs more neurons than other learning algorithms". Unfortunately, it is not supported by extensive research.

To summarise, our choice of using extreme learning machines for comparison was based on three factors:

- A common use of ELM as a baseline solution among random weights networks;
- A 1-to-1 correspondence between ELM and classical neural networks architecture;
- A scarcity of comparisons between ELM and classical neural networks.

1.2 Contribution

Our research was inspired by the need for practical neural network applications in classification and regression tasks. Training standard neural networks often requires computational power exceeding the capabilities of most research and development centres. The natural solution lies in alternative training methods. Because we did not find any comprehensive comparison between neither ELM or RVFL and classical networks, we wanted to objectively verify whether and to what extent ELM can be competitive for classical feedforward networks trained with backpropagation (with applied batch SGD) for demanding classification and regression tasks.

We performed the comparison twice. In the first series of experiments following the approach from [13], the network's and ELM's capacities (the number of neurons) are comparable. In this case, we compare both model properties in the context of training procedure. In the second one, we tried to reflect a machine learning practitioner approach to training models on a given dataset. This means that we tuned the model for a given dataset. For classical networks, we can make use of the extensive literature and publicly shared implementations. For ELM's, this is usually impossible, but short training times allow us to optimise hyperparameters efficiently. An essential premise to all experiments is to cover the vast and varied selection of datasets. This research limited the datasets to classification and regression tasks from public repositories (UCI [8] and University of Porto repository [46]), the current image classification benchmark datasets and artificially created tasks, skipping the text and audio domain datasets. We can summarise our contribution as follows:

- *comparison* of models' efficiency for more than 50 datasets using both training methods in terms of the achieved prediction quality and training times (in two scenarios: models with comparable capacity and well-suited (supported by literature) networks' architectures),
- *implementation* of both models available at: https://github.com/mkosturek/extreme-learning-vs-backprop,
- *statistical analysis* of the results, which ensure that any conclusions drawn from experiments are credible,
- *evaluation* of the decision boundaries of both models for several runs,
- ELM's hyperparameter *sensitivity analysis*,
- *qualitative comparison* focused on the practical application of both methods,
- *formulation* of postulates referring to the development direction for ELM for current demanding datasets.

The contributions are essential for machine learning researchers and practitioners to know which model to use depending on the dataset size, and what results can we expect regarding training time and the model's efficiency. Our research gives some insights in the decision boundaries of both models. It is crucial with the growing need to process massive datasets. The importance of our contributions can also be assistive to the random weights network researchers—by showing current challenges and obstacles in practical applications of ELM models and the need to compare enhanced ELM models to classical neural networks. It is also worth saying that ELM's hyperparameter sensitivity analysis is helpful in defining how to set their values properly.

1.3 Paper structure

The paper consists of six sections. Section 1 presents an introduction to the topic of this study. In Sect. 2, we characterise the two compared approaches to training neural networks—based on backpropagation in Sect. 2.1 and extreme learning in Sect. 2.2. In Sect. 3, we depict the procedure we adopted to conduct experiments. Section 5 presents experimental results and conclusions derived directly from them. In Sect. 6, we show our insights from the experimental study and present postulates about future development of ELMs. Finally, in Sect. 7, we summarise all conclusions drawn from this research.

2 Description of compared models

In this section, we briefly present both models describing their architectures and the method of training. We focus on classification and regression as they are the two most common tasks in real applications. It is necessary to prepare the training set used to find parameters θ (weights and biases) of the models to solve these problems. The training set consists of vector pairs $<\mathbf{x_i}, \mathbf{y_i}>$, i.e. the input vector and corresponding output vector (for classification outputs are encoded using one hot principle). In the case of regression, the corresponding output is a scalar y_i. Formally, the task is solved by a function $f(\mathbf{x}; \theta)$ implemented by a classical neural network or ELM. In the case of ELM, the parameters θ are calculated in one step, in opposite to classical network where training is an iterative procedure that optimises a loss function $\mathcal{L}(f(\cdot; \theta), \mathbf{x}, \mathbf{y})$. Having model

parameters, the output $\hat{\mathbf{y}}(\mathbf{x})$ specifies the predicted class or predicted value in the case of regression.

2.1 Classical neural networks

The integral part of each neural network is a neuron. The formal description of its operation is described by Eq. 1.

$$\hat{y}(\mathbf{x}) = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{1}$$

where $\hat{y}$ is the neuron's output, N is the number of inputs, x_i stands for the value of the ith input and w_i for the weight assigned to the ith input, b depicts the bias, and f is an activation function.

Each neuron has many inputs represented by a vector $\mathbf{x} = [x_1, \ldots, x_N]$. The input signals combined by weighted sum create the total neuron input activity. Weights $\mathbf{w} = [w_1, \ldots, w_N]$ are assigned to each neuron input connection and are tuned during the training process. Bias b is a weight assigned to additional input, where the signal is always set to 1. The output signal $\hat{y}$ is created on the basis of the total input transformation by the activation function f.

In this paper, we consider layered, feedforward network architectures. The first one is the *multilayer perceptron* (MLP), where neurons are fully connected in neighbouring layers. Further, we will also describe the *convolutional neural network* (CNN).

MLP is composed of an input layer, one or many hidden layers, and an output layer. The number of hidden layers and the number of neurons in each layer are hyperparameters of a network. The number of outputs depends on the problem solved by the network (classification or regression). For simplicity, in the description below we assume the simplest MLP network with one hidden layer. The signal processing in the network can be described as a sequence of matrix operations for all patterns in the dataset $\mathbf{X}$ given in Eq. 2

$$\mathbf{H}(\mathbf{X}) = f(\mathbf{X}\mathbf{W}^{(h)} + b^{(h)}) \tag{2}$$

where $\mathbf{W}^{(h)}$ is a matrix of hidden layer's parameters, $b^{(h)}$ is a vector of hidden layer's bias values, and f is an activation function. The network output is defined by Eq. 3.

$$\hat{\mathbf{Y}}(\mathbf{X}) = f(\mathbf{H}(\mathbf{X})\mathbf{W}^{(o)} + b^{(o)}) \tag{3}$$

where $\mathbf{W}^{(o)}$ and $b^{(o)}$ are a matrix of weights and vector of biases in the output layer.

There are many different activation functions, commonly used are: sigmoid, hyperbolic tangent, step function, ReLU [19, 30], linear function, and softmax in the output layer for classification problems.

Another type of neural network considered in the comparison is the *convolutional neural network* (CNN) [24]. In the convolutional layers, neurons are not fully connected. A given neuron is only connected to a defined subset of neurons in the subsequent layer. Moreover, weights assigned to these connections are shared between neurons of a single feature map of the convolutional layer. They are widely used in image processing because they learn to extract complex image features that serve the performed task the most. In each convolutional layer, one defines a *kernel* (or a *filter*)—matrix with assumed sizes, significantly lower than the image resolution. Filter values are tuned during training the network; they correspond to neurons' connections weights. A single convolutional layer may consist of multiple filters, each producing a separate feature map. While processing an image, convolutional filters are moved across the image stepwise by a constant number of pixels, and convolution operation is calculated. It defines the total activation for one neuron in a convolutional layer, Eq. 4.

$$y_{kij}(\mathbf{X}, \mathbf{W}) = \sum_{p=1}^{P} \sum_{q=1}^{Q} w_{kpq}\, x_{i+p,j+q} \tag{4}$$

where $\mathbf{X}$ is the input image, $\mathbf{W}$ denotes a matrix of convolutional filters (size: $K \times P \times Q$), K is the number of filters (feature maps), $P \times Q$ is the size of a single convolutional filter, and i, j are a single neuron's coordinates.

Similarly to MLP, convolution output values are processed by an activation function, and then pooling is applied. Its role is to decrease the size of a feature map. Usually it is implemented as max operation (*max-pooling*) or average (*average-pooling*) from the sliding window. Pooling allows a convolutional network to be more robust to small image rotations and translations.

Neural network training is performed as an optimisation task. We search for the cost (loss) function minimum. It defines the error the network makes approximating the target function $\mathcal{F}$. Most commonly used loss functions are: mean square error, binary cross-entropy, and categorical cross-entropy.

The primary method of neural network training is the gradient descent method that is shown in Algorithm 1. It requires a training dataset $\mathcal{D}$. In each iteration, the network assigns its output (line 2), and then the model parameters are corrected by a small value in the direction opposite to the gradient of cost function $\mathcal{L}$ (line 3). As an effect, the cost value decreases. The procedure lasts until a stopping criterion is not satisfied.

Long training may cause network overfitting; therefore, in practice, some other techniques are applied. One of them is *early stopping*. It relies on setting aside an additional validation dataset. In every iteration, it is used to monitor the cost value. The training is stopped when the validation cost increases.

Algorithm 1 Gradient Descent Method for neural network training

Input: $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$ - training data,
Output:$\mathbf{w}$ - final weights values
Require: $\mathcal{L}$ - the cost function minimised during training,
$\mathcal{D} = (\mathbf{X}, \mathbf{Y})$ - training data,
f - the transformation performed by a network trained with the weights $\mathbf{w}$,
α - learning coefficient
$\mathbf{w} \leftarrow$ initial randomly assigned weights
1: **repeat**
2: $\hat{\mathbf{Y}} \leftarrow f(\mathbf{X}, \mathbf{w})$
3: $\Delta\mathbf{w} \leftarrow \frac{\partial \mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})}{\partial \mathbf{w}}$
4: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \Delta \mathbf{w}$
5: **until** stopping criterion is false
6: **return w**

The gradient descent method requires a calculation of the loss function gradient value for all network parameters. For complex, multilayered network architectures it is a complicated task; therefore, to solve this problem the *Backpropagation* algorithm [26] is in everyday use. It assigns the loss function gradient for the last network layer, and then using the chain rule, it computes the gradient value for the weights in the immediately preceding hidden layer. For a network with more layers, analogously, the gradient in the nth layer is calculated by propagating the loss function gradient from $(n+1)$th layer. We can calculate the loss function gradient for the weights in the nth network layer using the following recursive definition.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_n} = f_{n-1}\ \delta_n \tag{5}$$

$$\text{where:} \quad \delta_i = \begin{cases} \left(\mathbf{W}_{i+1}\delta_{i+1} \odot f_i'\right)^\top & i < L \\ \left(\mathcal{L}' \odot f_L'\right)^\top & i = L \end{cases} \tag{6}$$

where $\mathcal{L}$ is the loss function, L denotes the total number of layers (and the index of the output layer), $\mathbf{W}_i$ is the weight matrix between $(i-1)$th layer and ith layer, f_i is the activation vector in the ith layer, $'$ denotes a value of the derivative (or a gradient), and the operator $\odot$ is the Hadamard product.

The method described above is in its simplest form. In common use is stochastic gradient [2], where in a single iteration, the network parameters are updated based on a small portion of the training dataset—a batch. Another improvement is using momentum. Other advanced methods apply optimisation methods of learning coefficient as, for instance, ADAM [22] method. A high level of generalisation is the aim of each machine learning model. Techniques that increase generalisation are called regularisation. The most popular ones are *L2* method and *Dropout*.

2.2 Extreme learning machines

ELMs are based on random projections of input feature space to the hidden features and then on linear regression as opposed to classical neural networks. ELM hidden connections can be weighted randomly, and there is no need to tune them.

The most straightforward ELM architecture is a feed-forward neural network containing a single hidden layer. It resembles the MLP architecture, but the weights between input and hidden layer are randomly assigned, and not tuned. They perform some random *black-box* transformation from input feature space to the hidden features. The parameters are tuned in the output layer only.

Extreme learning itself is not necessarily limited to this simple architecture. It could be incorporated in many well-known deep learning models. Huang et al. [17] propose a method based on extreme learning and CNNs to perform image classification. The model was called local receptive fields ELM (LRF-ELM). Local receptive fields are a general concept. It assumes that a single neuron is responsible for the aggregating signal from a specific input image region. However, usually for implementation purposes, LRF-ELM simplifies a network to a single convolutional layer with pooling and a fully connected output layer. Only the weights of the output layer are optimised. The convolutional layer acts as the random feature extractor. In other words, the values of all convolutional filters are chosen at random. There are two additional operations specific to LRF-ELM. The first one is orthogonalisation of filters after their initialisation. It aims to minimise the risk of producing redundant features by randomly initialised convolutional filters. The second one is a specific pooling method—square root pooling, given in Eq. 7, which enables the data dimensionality reduction and introduces nonlinearity in the network. This operation is crucial because there is no activation function after the convolutional layer in LRF-ELM.

$$\text{Square Root Pooling}(\mathbf{X}) = \sqrt{\sum_i^{D_1} \sum_j^{D_2} \mathbf{x}_{ij}^2} \tag{7}$$

where $\mathbf{X}$ is the input feature map (output of convolution layer) with size $D_1 \times D_2$. It is a matrix of x_{ij} elements.

In order to compute optimal output layer weights $\boldsymbol{\beta}$, Eq. 9 has to be solved—network outputs should match the expected values $\mathbf{Y}$ from the dataset. $\mathbf{H}$ represents a matrix of random features extracted from the training dataset $\mathbf{X}$.

For single hidden layer networks, they are computed according to Eq. 8.

$$\mathbf{H} = f(\mathbf{X}\mathbf{W_0} + \mathbf{b}) \tag{8}$$

$$\mathbf{Y} - \mathbf{H}\boldsymbol{\beta} = \mathbf{0} \tag{9}$$

where: $\mathbf{X}$, $\mathbf{Y}$,-training inputs and expected outputs, respectively, $\mathbf{W_0}$, $\mathbf{b}$-weights and biases in a hidden layer—both random,f-activation function.

To find the output layer weights $\boldsymbol{\beta}$ from Eq. 9, it is necessary to calculate the inverse of the matrix $\mathbf{H}$, as presented in Eq. 10, which can be computationally difficult for high-dimensional or large datasets. To address this issue, ELM authors proposed to use Moore-Penrose pseudoinverse, which leads to the final solution for the optimal weights, as shown in Eq. 11. It is equivalent to the least squares optimisation.

$$\boldsymbol{\beta} = \mathbf{H}^{-1}\mathbf{Y} \tag{10}$$

$$\boldsymbol{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y} \tag{11}$$

This algorithm is called extreme learning. It is also applied in classification layer in LRF-ELM, which training procedure is presented in Algorithm 2.

Algorithm 2 LRF-ELM training procedure

Input: Training data,

Output: Trained LRF-ELM model

1: Randomly initialise convolutional filter weights

2: Orthogonalise filter weights

3: Calculate a matrix of hidden features **H** (result of convolution and pooling)

4: Compute optimal weights for classification layer using Eq. 11

5: **return** LRF-ELM (with filters set in Step 2 and output weights from Step 4)

Just as with classical neural networks, there exist some techniques that improve the performance of ELMs. ELMs can be regularised using L2 regularisation [12, 17]. Equation 12 shows regularised solution for optimal ELM parameters. C denotes regularisation coefficient and $\mathbb{1}$ is the identity matrix.

$$\boldsymbol{\beta} = (C\mathbb{1} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y} \tag{12}$$

Even though ELM parameters are chosen at random, the probability distribution used for sampling them may impact the model performance. Authors of [45] compare various distributions with different variances. Authors recommend an excellent default choice of distribution—Gaussian with mean equal to 0 and a standard deviation less than or equal to 0.1.

The presentation of the essential elements of both models and teaching methods was the aim of this section. The next one is devoted to experimental research to compare both models in terms of learning time and achieved results.

3 Research methodology

The experiments on ELM hyperparameter sensitivity allow us to know hyperparameters' influences on the final ELM's results. This analysis is described in A. It enables us to assess which parameter has a particularly strong impact on the model responses and to determine good default values for hyperparameters.

We have designed two series of experiments: in the first one, models have comparable capacities. In the second one, they have well-suited architectures based on the literature review or hyperparameter optimisation.

The first series is designed to compare just the learning algorithms. Given identical network architectures, we train them using extreme learning and backpropagation. This comparison can show whether it is beneficial to choose one algorithm over the other. This series consists of two parts. The first part refers to ELM and fully connected networks trained with backpropagation, performing classification and regression tasks on data with vector representation. This part of experimentation utilises well-known, real-life benchmark datasets, datasets used in [16] and several artificial datasets described in Sect. 4.

To ensure the objectivity of conclusions, we used datasets with diverse properties concerning data dimensionality, class balance, the sparsity of features, presence of continuous and discrete features. Datasets were acquired from two public repositories: UCI [8] and University of Porto repository [46]. Tables 1 and 2 present characteristics of datasets for classification and regression.

The second part of this series is devoted to the image classification task. We considered two approaches to image classification: • using external extractor of visual features and fully connected network, • using convolutional network, which learns features on its own.

Previously mentioned LRF-ELM models [17] allow researchers to utilise extreme learning for convolutional networks. Authors of [18] show that ELM for image classification performs well when using HOG features (histogram of oriented gradients). The paper considers only one task—road signs classification. However, it seems worthy to investigate this approach on more datasets, described in Table 5. Therefore, the models covered in this comparison are:

Table 1 Characteristics and descriptions of datasets for regression from UCI and University of Porto repositories used for comparisons

Dataset	No of samples	No of features	Min. value of target variable	Max. value of target variable	Description
Auto MPG	392	21	9.0	46.6	Estimation of car fuel consumption. Five input variables are continuous, remaining 16 features represent one-hot encodings of two discrete variables
Bank	8192	8	0.0	0.8	Estimation of the number of customers who decide to quit the queue
California Housing	20,640	8	0.15	5.0	Median prediction of housing prices in a neighbourhood described by 8 input variables. Dataset acquired from StatLib repository [29]
Delta Ailerons	7129	5	$-2.1 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$	Prediction of variance of ailerons positioning in the F-16 fighter aircraft
Machine CPU	209	6	6	1150	Prediction of multiple CPUs performance based on their specifications
Servo	167	19	0.13	7.1	Prediction of latency in a servo system

Table 2 Characteristics and descriptions of datasets for classification from UCI repository used for hyperparameter sensitivity analysis and for comparisons

Dataset	No of classes	No of samples	No of features	Minority class size	Majority class size	Description
Breast Cancer (Ljubljana)	2	277	41	81	196	Prediction of a breast cancer recurrence. All features are one-hot encoded
Breast Cancer (Wisconsin)	2	569	30	212	357	Prediction of a breast cancer class (benign or malignant) [28]
CNAE-9	9	1080	856	120	120	Documents represented with bag-of-words; sparse and high-dimensional encoding, classes are equinumerous
Forest Cover Type	7	581,000	54	2747	283,000	Forest classification based on cartographic features; large dataset with imbalanced classes
Glass Identification	6	214	9	9	76	Classification of glass shards collected as crime evidence. Highly imbalanced classes
Landsat	6	6435	36	626	1533	Land cover classification using satellite image continuous features. Classes are imbalanced
Sonar	2	208	60	97	111	Land mines and stones classification based on a sonar spectrogram; high dimensionality-to-size ratio, classes balanced
Urban Land Cover	9	675	147	29	122	Land cover recognition using features from aerial images; relatively small, highly dimensional
Wine	3	178	13	48	71	Classification of 3 types of wine

- LRF-ELM,
- Convolutional network trained with backpropagation,
- ELM using HOG features,
- fully connected network trained with backpropagation on HOG features.

In the second series, we try to tune the architectures and hyperparameter set-ups for a given problem, just like a machine learning practitioner would do. In this task, we make use of the available literature. This means that for classical network we often refer to the best found results reported in other papers, while we perform our own hyperparameter search for ELMs. This series also consists of two parts. Analogously, the first part is based on classification and regression on data with vector representation (this time however, we do not use synthetic datasets), and the second concerns image classification. The choice of image datasets is extended to two variants of ImageNet—with resolutions of 16×16 and 32×32 pixels. We compare training times when possible—for training performed on our own, and when the time is given in the papers we refer to. But the main focus lies on the comparison of the performance measured with various metrics:

- F1-score in the first series of comparison for the classification tasks. In this series, we performed training and evaluation of all models on our own. Therefore we were able to measure the F1-score, which is more robust to imbalanced classes than accuracy.

- Accuracy in the first series of comparison for the classification tasks. This time we referred to results reported in literature which are usually stated using only the accuracy metric.
- RMSE (normalised) for the regression task.

Their definitions are given below.

F1-score. It is a measure of binary classification quality. It summarises the number of Type I and Type II errors made by the classifier. Type I errors are also called *false positives*. They correspond to the situations where the classifier mistakenly predicts a positive class. Type II errors *false negatives* mean that the classifier mistakenly predicted the negative class. F1 score is therefore defined as stated in Eq. 13

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{13}$$

where: TP- number of *true positives* (correct predictions of positive class,), FP- number of *false positives*, FN- number of *false negatives*. To use this metric for multiclass classification, we can simply compute binary F1-score for each class separately and then evaluate an average F1-score weighted by the numbers of samples in each class. The formal description is given in Eq. 14.

$$F_1 = \frac{\sum_{k=1}^{K} w_k F_1^{(k)}}{\sum_{k=1}^{K} w_k} \tag{14}$$

where: $F_1^{(k)}$- F1 score for class k, K- number of classes, w_k- number of samples in class k. For simplicity, we will denote both binary and multiclass version of this measure as F1 score.

Accuracy. It is the most basic measure of classification performance. It is simply a ratio of correctly classified samples and the number of all data samples in the dataset, as stated in Eq. 15.

$$Accuracy = \frac{\#of\ correct\ predictions}{\#of\ all\ test\ data\ samples} \tag{15}$$

RMSE. Root mean square error is a measure of regression quality. It is defined in Eq. 16.

$$\mathrm{RMSE}(\hat{\mathbf{Y}}, \mathbf{Y}) = \sqrt{\frac{\sum_i^N (\hat{y_i} - y_i)^2}{N}} \tag{16}$$

where: $\mathbf{Y}$-vector of expected model responses for N data samples, $\hat{\mathbf{Y}}$-vector of actual model responses for N data samples, N-number of samples in the testing dataset. In this paper, we present normalised values of RMSE. Normalisation is performed by dividing the value of RMSE by the range of the target variable (the difference between the greatest and the lowest value). It makes a more straightforward interpretation of the model error magnitude without any domain knowledge about the target variable.

4 Datasets used

In selecting datasets for experimental research, we were guided by the diversity of the data in terms of class balancing, data dimensions, number of classes, continuous and discrete data, and sparsity of features. Datasets were acquired from two public repositories: UCI [8] and University of Porto repository [46]. Table 1 presents characteristics of datasets for regression and Table 2 for classification. Moreover, some synthetic datasets were considered.

We also propose three types of artificial classification datasets and several datasets for regression. They are characterised in Tables 3 and 4, and described in detail in Appendix B.

Image classification datasets. None of the available ELM studies presents results obtained on benchmark image classification datasets, such as MNIST OCR and CIFAR-10. For this reason, we included them in the experiments. In addition, we use the NORB dataset, which was used in [17], and the GTSRB set, used in [18]. All image datasets are characterised in Table 5.

5 Experimental study

We conducted all experiments on a computer with a specification presented in Table 6.

For the needs of the study, the ELM and LRF-ELM models were implemented using Python 3.6 and `PyTorch` 1.4 library [34]. All the reference models (fully connected and convolutional networks with backpropagation) were also implemented based on the following libraries: `scikit-learn`, `scipy`, `scikit-posthocs`, `OpenCV`. Our implementation is available at: github.com/mkosturek/extreme-learning-vs-backprop.

The first experiment compares ELM models and neural networks trained using backpropagation (also referred to as MLP) in each series. The next one focuses on image datasets recognition. The simple ELM and MLP models working on HOG features extracted from images are compared. We also examine models dedicated to image classification: LRF-ELM and convolutional neural networks.

5.1 Models with comparable capacity

In this series, we compare models with the same architecture and comparable number of neurons.

Table 3 Characteristics of artificial datasets for classification used for hyperparameter sensitivity analysis and for comparisons

Dataset	No of classes	No of samples	No of features	Minority class size	Majority class size
Nested hyperspheres	2	2000	2–1000	50%	50%
Hypercube vertices	2	2000	2–10	~50%	~50%
Intertwined spirals 2D	2	2000	2	50%	50%
Intertwined spirals 3D	2	2000	3	50%	50%

Table 4 Characteristics of artificial datasets for regression used for comparisons

Dataset	No of samples	No of features	Min. value of target variable	Max. value of target variable
Linear function	2000	1–1000	0	0
Second degree polynomial of multiple variables	2000	1–1000	0	0
Sine and cosine	2000	1	−1	1
SinC	2000	1	−0.2	1
Friedman spline I	2000	5–1000	~0	~30
Friedman spline II	2000	4	~0	~1760
Friedman spline III	2000	4	~0	~1760

Table 5 Specifications and descriptions of image classification datasets used in this study

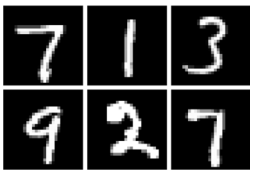
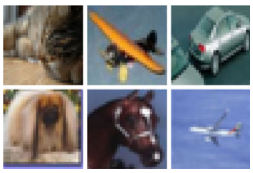
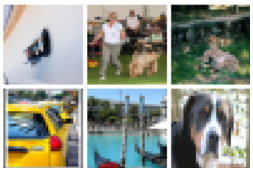
Dataset	Size	Images examples	Description
MNIST OCR	70,000		Recognition of handwritten digits. **Image resolution**: 28×28 px, Black and white images, **Number of classes**: 10
CIFAR-10	60,000		Recognition of 10 various classes of objects on photographs. **Image resolution**: 32×32 px, RGB colour images, **Number of classes**: 10
Small NORB	48,600		Classification of objects photographed from various angles and with varying lighting. **Image resolution**: 96×96 px, Grayscale images, **Number of classes**: 5
GTSRB	51,839		Recognition of German road signs. **Image resolution**: various, we rescaled them to the median resolution: 43×43 px, RGB colour images, **Number of classes**: 43
ImageNet	1.2 mln		Classification of various objects. **Image resolution**: 224×224, rescaled to two variants: 16×16 px and 32×32 px, RGB colour images, **Number of classes**: 1000

Table 6 Specification of the computer on which the experiments were conducted

Parameter	Value
Processor (CPU)	Intel i7-4790K
Memory size	24 GB
Graphical processor (GPU)	NVidia AORUS GeForce GTX 1080 Ti 11G
Operating system	Ubuntu 18.04

5.1.1 Experiment 1: comparison of ELM and neural networks performance and training times for input data with vector representation

The purpose of the study is to conduct a comparison of the performance and training time of ELM models and neural networks characterised by similar capacities using datasets with the vector representation for regression and classification tasks. We based the choice of values of 3 hyperparameters (regularisation coefficient, activation function, and hidden layer size coefficient) on the hyperparameter sensitivity analysis (Appendix A). We chose values that resulted in decent performances on average on various datasets. Eventually, the following hyperparameter values were applied:

Regularisation - L2 was applied for both ELM and MLP with $C = 0.01$,
Activation function - ReLU for both MLP and ELM,
Optimisation algorithm - ADAM algorithm [22] was used while training by backpropagation,
Batch size - set to 16 while training by backpropagation,
Learning rate - set to 0.001 while training by backpropagation,
Training stopping criteria - early stopping with patience set to 3 epochs and maximum number of epochs set to 200,
Hidden layer size coefficient - based on the analysis of the hyperparameter sensitivity in A, $h = 10$. It applies for both ELM and backpropagation trained network. (It is in contrast to the comparison presented in [16], where the number of neurons in ELM was usually about twice as high as the MLP).
Significance level - $\alpha = 0.05$

The experiment was conducted on 23 classification datasets and 30 datasets for regression presented in Tables 1 and 2, respectively. Moreover, synthetic datasets were used. They are described in detail in Appendix B. We generated data in multiple variants with different dimensions, varying from 1 up to 1000 dimensions. Each synthetic dataset contained 2000 examples.

Each model was evaluated on every dataset using a fivefold cross-validation, repeated 10 times, giving a sample of 50 results for each model on each dataset. In the classification task, the F1 measure was used to compare the performance. For the regression task, we used normalised RMSE. Normalisation was performed by dividing the RMSE measure by the range of the target variable (the difference between the maximum and minimum). To evaluate the training speed, we measured the training time in seconds. To ensure the reliability of the comparison of the average values, we used the dependent *t*-test for paired samples, but, as indicated in [3], in the case of repeated cross-validation, the basic form of this t-test may lead to false and non-reproducible results. Therefore we used a modified paired-samples *t*-test, called *corrected repeated k-fold cross-validation test*.

Tables 7 and 8 show the results of comparison of performance and training times in classification and regression tasks. Whenever a *p*-value from the statistical test was below the significance level $\alpha = 0.05$, the better model's result was presented in bold, denoting a significant difference between compared models.

Discussion Considering classification performance in terms of F1 measure, the results presented in Table 7 show that for most sets (15 out of 23) the statistical test did not show any significant difference between the average results of the models. On 8 sets, where the differences were statistically significant, better results were achieved by ELM. It is worth noting that the differences between the models were found to be significant on high-dimensional synthetic sets rather than on their low-dimensional counterparts.

A qualitative comparison of decision regions of sample ELM and MLP models is also interesting. For the simplest, two-dimensional datasets, they are visualised in Figs. 1, 2, 3. It is worth underlining the significant differences in the shapes of the decision regions for different runs of the ELM model.

Although the identified decision regions usually correctly cover the training set's observations, these areas undergo significant shape changes just beyond the concentration areas of observations in the case of ELM. It is particularly noticeable for a dataset of nested hyperspheres, where the inner circle class region also appears outside both circles. Decision boundaries for MLP seem to be much simpler than for ELM. It can be seen that the decision boundaries for ELM vary for each of the 15 runs of the method. This phenomenon suggests better generalisation

Table 7 Comparison of the average F1-measure and training times (in seconds) for ELM and MLP

Dataset	F1-measure		Training time	
	ELM	MLP	ELM	MLP
Breast Cancer (Ljubljana)	0.4722	0.4755	**0.0010**	0.1013
Breast Cancer (Wisconsin)	0.9729	0.9699	**0.0015**	0.4296
CNAE-9	0.9261	0.9306	**0.1278**	28.5683
Forest Cover Type	0.7140	0.6395	**2.1873**	292.6776
Glass	**0.6418**	0.5197	**0.0004**	0.2415
Landsat	**0.8613**	0.7762	**0.0104**	1.7258
Sonar	0.7767	0.7387	**0.0009**	0.1666
Urban Land Cover	0.7824	0.7559	**0.0075**	0.6181
Wine	0.9787	0.9820	**0.0004**	0.3228
Hypercube vertices (2 dim.)	1.0000	1.0000	**0.0008**	1.1139
Hypercube vertices (3 d.)	1.0000	1.0000	**0.0004**	1.9344
Hypercube vertices (5 d.)	**0.9859**	0.9545	**0.0004**	3.7209
Hypercube vertices (7 d.)	0.7575	0.7140	**0.0006**	1.9322
Hypercube vertices (10 d.)	**0.5672**	0.4536	**0.0011**	0.3202
Hyperspheres (2 d.)	0.7828	0.9154	**0.0003**	5.5755
Hyperspheres (3 d.)	0.9144	0.9142	**0.0004**	4.3765
Hyperspheres (10 d.)	**0.9272**	0.7180	**0.0009**	1.6621
Hyperspheres (30 d.)	**0.9362**	0.4153	**0.0040**	0.2685
Hyperspheres (100 d.)	**0.8683**	0.4789	**0.0312**	0.6166
Hyperspheres (300 d.)	**0.6671**	0.3035	**0.1288**	3.3525
Hyperspheres (1000 d.)	0.5480	0.4274	**0.4061**	46.9665
Spirals (2 d.)	0.6960	0.6655	**0.0004**	0.7449
Spirals (3 d.)	0.6195	0.5705	**0.0004**	0.5910
Number of outcomes with results significantly better than the competing model's	8	0	23	0

When p-value from statistical test is below the significance level $\alpha = 0.05$, indicating significant difference in models' average performance, the better result is presented in bold

capabilities of MLPs. For most regression datasets, the statistical test showed significant differences in the quality of model performance (22 out of 30 sets).

Of these 22 comparisons, only 8 indicated better results of ELM model, and 14 indicated better results on MLP. This effect is different from the classification comparison result. However, in Table 8, it can be seen that the MLP achieved significantly better results on only one real-life set, *Servo*, and three synthetic sets: *linear function*, *2nd degree polynomial*, *Friedman I*, but with a majority of their possible dimensions. ELM proved to be significantly better on three real sets, on synthetic sets based on trigonometric functions and Friedman II and III functions. It should be noted that all of these 8 sets are low-dimensional, e.g. dimensionality did not exceed 10.

The obtained results indicate that ELM models achieve better results in low-dimensional and nonlinear regression problems, which occur in many real-life tasks, in sets based on trigonometric functions and Friedman II and III curves based on physical phenomena. MLP, on the other hand, achieves better results for high-dimensional problems and those with linear or polynomial dependencies.

The comparison shows that ELM models are significantly faster than neural networks trained by backpropagation. On each of the 53 sets, for both classification and regression tasks, the statistical test showed a significant difference in processing times, always in favour of ELM. Typically, learning times for MLP turn out to be 4 orders of magnitude higher than for ELM.

5.1.2 Experiment 2: comparison of ELM and neural networks performance in image classification

The aim of the experiment is to compare networks trained with extreme learning and backpropagation in the task of image classification. We compare F1-measure and training times. The following hyperparameter values were set after preliminary studies:

Table 8 Comparison of the average RMSE and training times (in seconds) for ELM and MLP

Dataset	RMSE		Training time	
	ELM	MLP	ELM	MLP
Auto MPG	0.0784	0.0746	**0.0010**	0.9232
Bank	**0.0474**	0.0796	**0.0028**	1.4651
California Housing	0.1456	0.1494	**0.0072**	5.5622
Delta Ailerons	**0.0394**	0.0716	**0.0012**	1.1028
Machine CPU	**0.0409**	0.0813	**0.0003**	0.6753
Servo	0.1247	**0.0795**	**0.0006**	0.2217
Linear function (1 dim.)	0.0107	0.0078	**0.0004**	2.1336
Linear function (3 d.)	0.0021	0.0012	**0.0005**	1.0053
Linear function (10 d.)	0.0037	**0.0018**	**0.0012**	1.6946
Linear function (30 d.)	0.0056	0.0046	**0.0040**	2.2008
Linear function (100 d.)	0.0099	**0.0050**	**0.0317**	1.4333
Linear function (300 d.)	0.0216	**0.0063**	**0.1311**	6.3831
Linear function (1000 d.)	0.0366	**0.0053**	**0.4212**	138.9691
2nd deg. polynomial (1 d.)	0.1214	**0.0078**	**0.0003**	3.0669
2nd deg. polynomial (3 d.)	0.0721	**0.0061**	**0.0004**	6.1652
2nd deg. polynomial (10 d.)	0.0872	**0.0073**	**0.0010**	7.1019
2nd deg. polynomial (30 d.)	0.1202	0.1056	**0.0042**	4.7240
2nd deg. polynomial (100 d.)	0.2132	**0.0010**	**0.0312**	0.6630
2nd deg. polynomial (300 d.)	0.2600	**0.1011**	**0.1291**	2.9807
2nd deg. polynomial (1000 d.)	0.2297	**0.0544**	**0.4217**	44.3745
Sin	**0.0570**	0.2078	**0.0010**	3.1900
Cos	**0.0522**	0.2314	**0.0013**	4.5293
SinC	**0.0608**	0.1303	**0.0013**	0.7917
Friedman I (5 d.)	0.0596	**0.0423**	**0.0014**	4.5284
Friedman I (30 d.)	0.0963	0.0813	**0.0047**	1.5832
Friedman I (100 d.)	0.1326	**0.0974**	**0.0330**	1.2921
Friedman I (300 d.)	0.1651	**0.1073**	**0.1353**	7.2085
Friedman I (1000 d.)	0.1443	0.1401	**0.4193**	107.4185
Friedman II	**0.0165**	0.0405	**0.0011**	7.6029
Friedman III	**0.0768**	0.0949	**0.0010**	1.1100
Number of outcomes with results significantly better than the competing model's	8	14	30	0

When p-value from statistical test is below the significance level $\alpha = 0.05$, indicating significant difference in models' average performance, the better result is presented in bold

Regularisation coefficient - $C = 0.01$. It was set based on the preliminary experiment, which is not shown here.
Hidden layer size coefficient - for models based on HOG: $h = 10$,
Activation function - *ReLU*,
Architecture of convolutional networks - one convolutional layer, one pooling layer, with no activation function for both CNN and LRF-ELM. A flattened feature map after pooling is the input to the fully connected classification layer,
Number and size of convolutional filters - 10 filters, with a size of 5×5 pixels, the stride is 1 pixel for both LRF-ELM and CNN,
Pooling method and window size - Square Root Pooling for LRF-ELM and CNN. Window size is 5×5 pixels, with the stride equal to 2 pixels,
Optimisation algorithm when using backpropagation - ADAM [22],
Batch size while training by backpropagation - for MLP model with HOG features: 32; for CNNs, the batch size was 128,

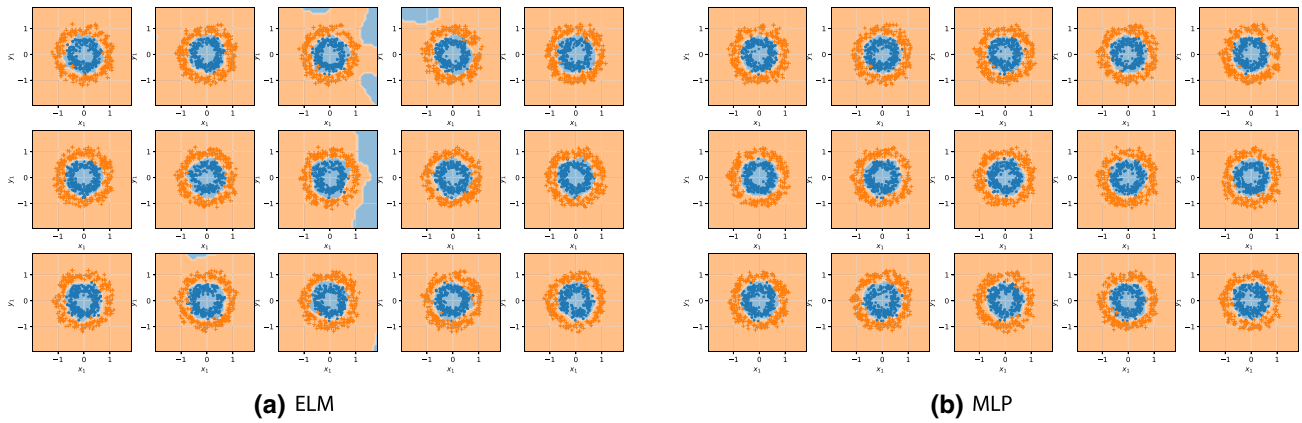


Fig. 1 Visualisation of decision regions of sample models on the nested circles dataset (two-dimensional hyperspheres)

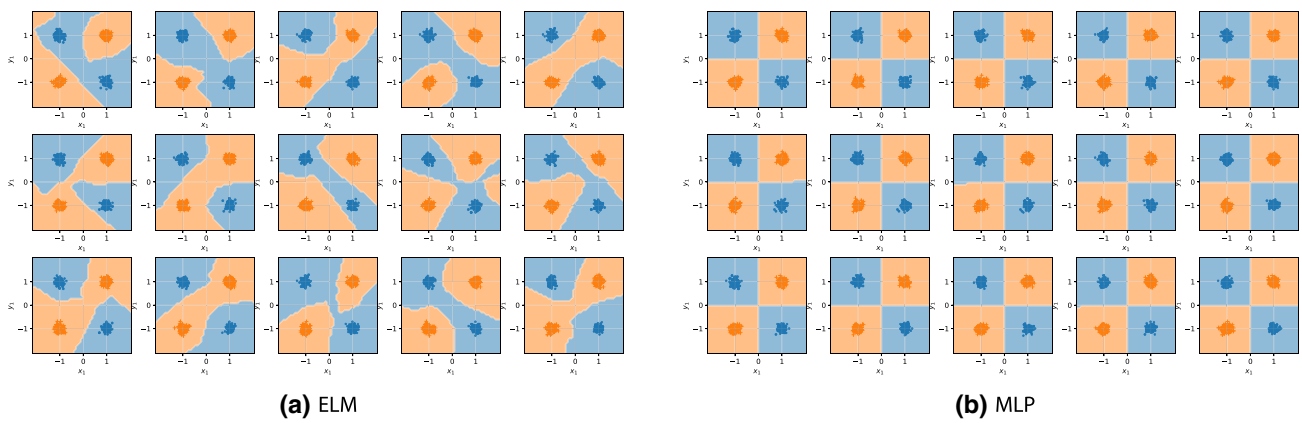


Fig. 2 Visualisation of decision regions of sample models on the XOR classification dataset (two-dimensional hypercube vertices)

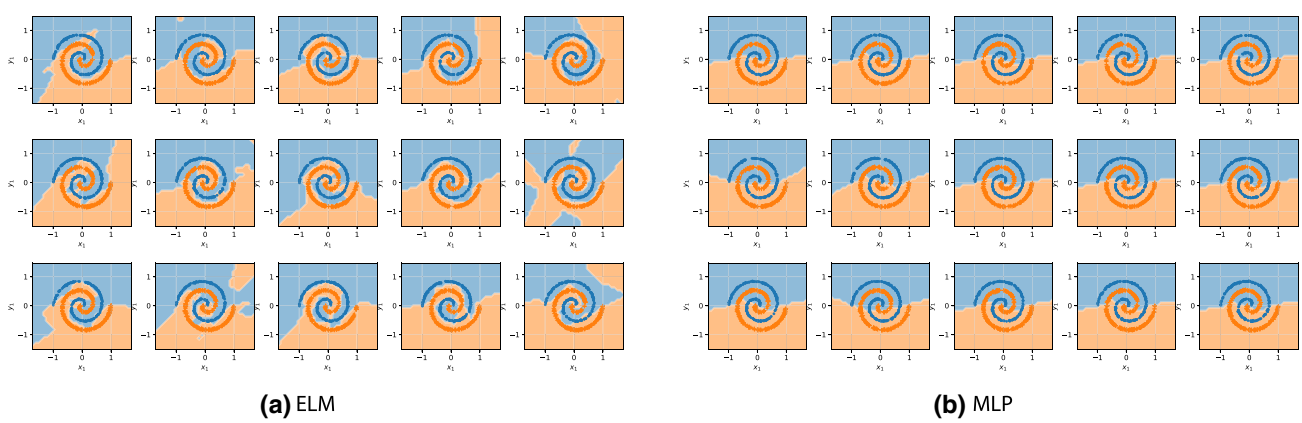


Fig. 3 Visualisation of decision regions of sample models on the intertwined spirals dataset

Learning rate - set to 0.001 while training by backpropagation,
Training stopping criteria - early stopping with patience set to 3 epochs and maximum number of epochs set to 200,
Significance level for statistical testing - $\alpha = 0.05$,
HOG descriptor set-up - the number of gradient orientations is 9, the block size is 2×2 cells, each cell is 8×8 pixels.

The experiment was conducted on 4 datasets, the characteristics of which are presented in Table 5. *CIFAR-10* dataset was used in its original form, the HOG vector for

this set consists of 324 features. *GTSRB* dataset consists of various sizes of images therefore, for the LRF-ELM models and convolutional networks, all images were scaled to 32×32 pixels. The HOG features were calculated on images scaled to a resolution of 43×43 pixels. This size is a median size of all images in the set. The HOG vector for this set consists of 576 features. *MNIST* dataset was used in its original form. The vector of the HOG features for this set consists of 144 features. All images in the *NORB* dataset are 96×96 pixels. Due to hardware limitations, for networks using convolutional layers, the images were scaled to a resolution of 32×32. HOG features were determined for images scaled to 48×48 pixels. The HOG vector for this set consists of 900 features.

On every dataset, a fivefold cross-validation was repeated five times for each of the tested models. Comparison is then based on performance measures averaged over these repetitions. For performance comparison, F1 measure was used, while the time measured in seconds was used to assess the training speed. Time measurement includes only model training and not HOG features extraction.

Figure 4 shows the results of the performance comparison for each of the four datasets. We performed the corrected repeated cross-validation t-test pairwise for all models, we assumed significance level $\alpha = 0.05$. These tests allowed us to determine that on CIFAR-10 and GTSRB datasets, both CNN and HOG+ELM approach resulted in the best performances without significant differences between them; on MNIST dataset, CNN was solely the best performing model; on NORB the best results were achieved with HOG+ELM approach. Moreover, these tests show that the performance results of CNN models trained on CPU and GPU do not differ, so they are only used to compare training times.

Figure 5 shows a comparison of the training times. Corrected repeated cross-validation tests were also performed. They showed that each of the average training times was significantly different from the others.

Discussion Figure 4 shows that CNN and ELM models using HOG features provide the best classification performance the same number of times. On CIFAR and GTSRB sets, these two models achieved results that did not differ significantly. On the MNIST set, convolutional networks using backpropagation proved to be better, and on the NORB set, the ELM model achieved the best results.

LRF-ELM achieved a result comparable to CNN only on NORB and GTSRB, while in the two other cases networks trained by backpropagation were superior.

A surprisingly high improvement in classification performance was achieved by the use of extreme learning instead of backpropagation when using HOG features. Three out of four ELM models using these features also proved superior to the LRF-ELM model, which has been designed for image processing.

It should be noted that the architecture of the convolutional networks used in this experiment was very limited in comparison with the usual design of such networks; for example, the number of filters was limited to 10. This is due to the high memory requirements of the LRF-ELM model and hardware limitations. In the case of convolutional networks with backpropagation, the size of the network could be greatly increased. Therefore, drawing conclusions from a comparison of image-based models with those using HOG features may not be fully justified. The superiority of a simple extractor over the use of a

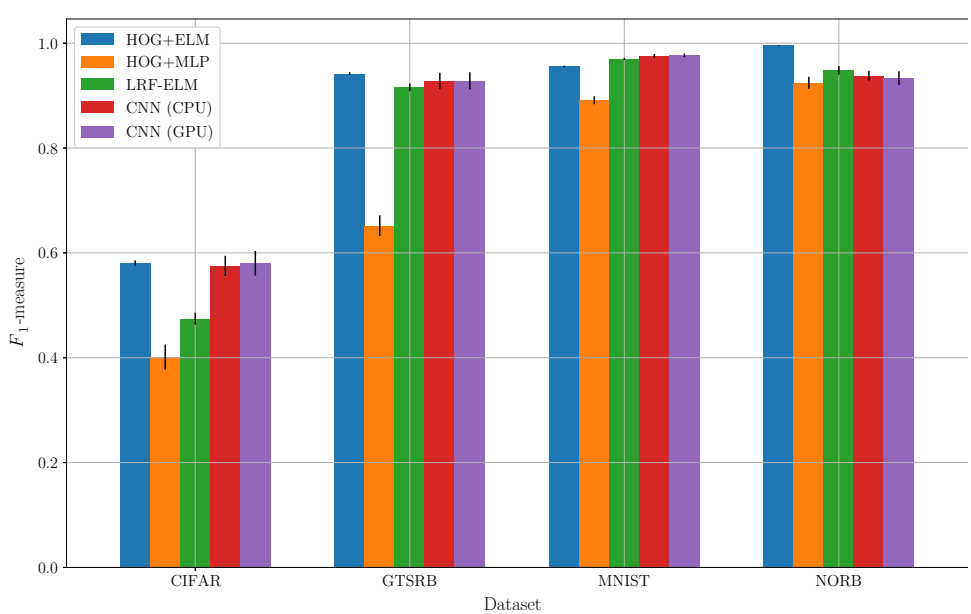


Fig. 4 Comparison of the classification performance of tested models on 4 datasets for image classification. Average and standard deviation of results obtained from five repetitions of fivefold cross-validation are presented

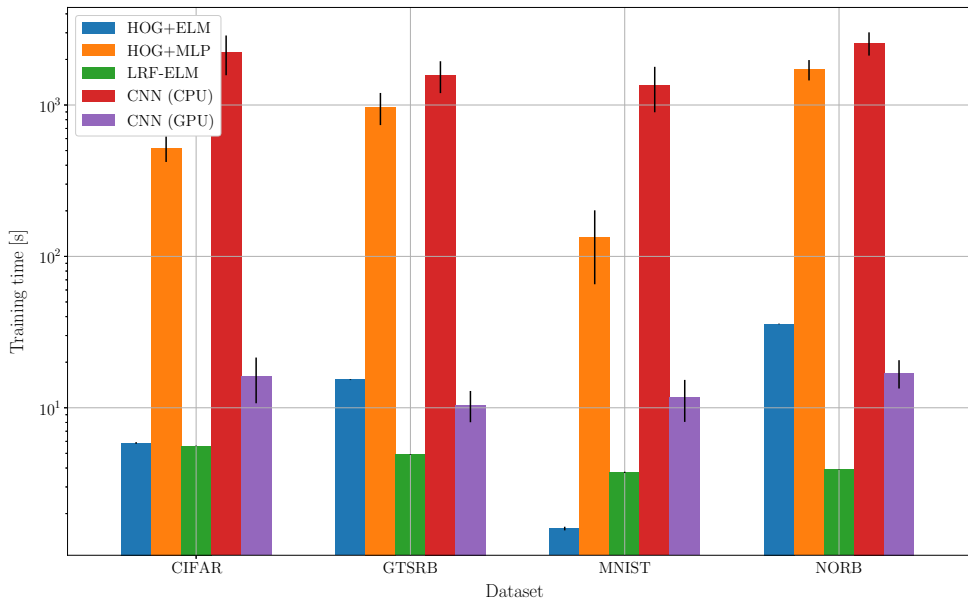


Fig. 5 Comparison of training times of the tested models on 4 image classification datasets. Average and standard deviation of results from five repetitions of fivefold cross-validation are presented. Times are shown on a logarithmic scale due to their considerable range

dedicated image network architecture cannot be undeniably confirmed.

Again, the comparison showed that extreme learning is significantly faster than backpropagation. On three out of four sets, LRF-ELM proved to be the fastest, while ELM using the HOG feature extractor was the fastest on a single set. The use of the GPU allowed to accelerate backpropagation training more than a hundred times to a level comparable to ELM models using HOG features. However, it did not achieve learning times as short as LRF-ELM.

5.2 Models with well-suited architectures for a given dataset

In this series, we conduct experiments using machine learning practitioner's approach. Following this assumption, the experiments compare both methods' performance using models with hyperparameter values tuned to achieve the best possible results or, if available, using reference to state-of-the-art results.

5.2.1 Experiment 1: comparison of ELM and neural networks performance and training times for input data with vector representation

In this scenario, the goal is to find a solution that performs the best. Therefore we utilise the hyperparameter search for ELMs and make use of works published throughout the years of classical networks development. Then we compare the best solutions found on datasets for regression and classification tasks with vector representation. Tables 9 and 10 show the best ELM hyperparameter set-ups found for each dataset. Due to some discrepancy in literature concerning metrics used for evaluating regression models, we decided to perform hyperparameter search for classical networks on our own. Table 11 shows the best hyperparameters found for each regression dataset. Tables 12 and 13 present the best results achieved with ELMs and classical networks.

Discussion This experiment shows that both extreme learning and backpropagation are able to produce a model best fitting a given task. However, classical networks proved to be superior on majority of tested datasets in both classification (6 out of 9 datasets) and regression (4 out of 6 datasets). It must be noted nonetheless that there are some cases when ELMs go behind standard networks only by a small margin (e.g. Machine CPU and Auto MPG datasets) and when they surpass classical networks significantly (e.g. Sonar and Bank datasets).

Overall, this experiment demonstrates that the many years of extensive research on backpropagation-based networks were beneficial. Classical networks are a solid, robust framework allowing very efficient modelling. However, ELMs seem to be a good complement for classical networks—easy and quick to train, having a potential of producing very competitive results. Considering the disproportion in the amount of research on ELMs compared to backpropagation, extreme learning seems to be promising approach, which is worth further development.

5.2.2 Experiment 2: comparison of ELM and neural networks performance in image classification

Just like in the previous experiment, the aim of this study is to achieve the best possible performance with each method in the image classification task. This comparison covers a

Table 9 Hyperparameter set-up for the best found ELM models for each classification dataset

Dataset	Hidden layer size	Activation function	Regularisation coefficient
Breast cancer (Wisconsin)	3000	ReLU	10
Breast cancer (Ljubljana)	300	Threshold	10
Wine	100	Sigmoid	0.1
Urban land cover	10,000	Threshold	0.1
Sonar	1000	Threshold	0.1
Glass	1000	Tanh	0.01
Landsat	10,000	ReLU	0.3
CNAE-9	10,000	Tanh	10
Forest cover type	3000	ReLU	1

Table 10 Hyperparameter set-up for the best found ELM models for each regression dataset

Dataset	Hidden layer size	Activation function	Regularisation coefficient
Servo	10	Tanh	10
Machine CPU	10,000	Tanh	0.1
Auto MPG	100	Tanh	0.003
Delta ailerons	10	ReLU	0.001
Bank	10	ReLU	0.001
California housing	300	ReLU	10

Table 11 Hyperparameter set-up for the best found MLP models for each regression dataset

Dataset	Hidden layer size	# of layers	Batch size	Early stopping patience	Activ. func.	Regul. coeff.
Servo	256	4	32	8	ReLU	0.001
Machine CPU	256	8	32	8	ReLU	10
Auto MPG	32	4	16	8	ReLU	0.1
Delta Ailer.	256	1	32	4	ReLU	0.001
Bank	256	1	16	8	ReLU	0.01
California H.	128	4	32	8	ReLU	0.01

large-scale dataset—ImageNet. It was constrained to two variants, with 16×16 and 32×32 resolution, due to the high memory complexity of ELMs. Again, we performed hyperparameter search for ELM-based classifiers and supported the comparison with results achieved with classical networks reported in the literature. Table 14 presents the best hyperparameter set-ups found for LRF-ELM models for each dataset. Table 15 shows the performance and time comparison of ELMs and backpropagation-based models.

In this experiment, we modified the preprocessing steps on the NORB dataset to match [37] to whom we compare our ELM's results.

Discussion Comparing current state-of-the-art deep learning models for image classification to the LRF-ELM clearly demonstrates classical deep learning superiority. LRF-ELM was inferior on every dataset considered, often by a great margin. This outcome was expected because LRF-ELM corresponds to a very simple CNN, whereas modern deep learning architectures are significantly more complex. In the case of image classification, it is hard to argue that the training speed is an advantage of ELMs, because the performance decrease is too high. Presently, extreme learning is not a competitive alternative for deep learning-based image classification.

Again, this experiment shows a high disproportion in the development of extreme learning-based architectures compared to backpropagation. Extending ELMs into a framework allowing efficient implementation and training of more complex network architectures could render valuable.

6 Findings based on the performed research

As mentioned before, extreme learning machines are highly memory-consuming. This is because of the need to compute the inverse of the latent representation of the

Table 12 Comparison of the accuracy for the best found ELM and MLP solutions for given classification datasets

Dataset	Model	Accuracy (%)
Breast cancer wisconsin)	ELM (own)	93.02
	MLP [48]	**96.30**
Breast cancer (Ljubljana)	ELM (own)	71.43
	MLP [48]	**73.50**
Wine	ELM (own)	96.30
	MLP [48]	**98.30**
Urban land cover	ELM (own)	**87.25**
	MLP (own)	83.33
Sonar	ELM (own)	**84.38**
	MLP [48]	76.40
Glass	ELM (own)	54.55
	MLP [48]	**68.70**
Landsat	ELM (own)	89.23
	MLP [44]	**91.00**
CNAE-9	ELM (own)	**98.15**
	CoPACoRSET-MLP [35]	95.95
Forest cover type	ELM (own)	77.75
	MLP [27]	**84.78**

For each dataset, the result obtained by the better model is presented in bold

Table 13 Comparison of the RMSE (normalised) and training times for best found ELM and MLP solutions for given regression datasets

Dataset	Model	RMSE	Training time[1]
Auto MPG	ELM (own)	0.0751	1.5 min
	MLP (own)	**0.0692**	1.5 h
Bank	ELM (own)	**0.0000**	19 min
	MLP (own)	0.0692	7.5 h
California housing	ELM (own)	0.1627	1 h
	MLP (own)	**0.1307**	20 h
Delta ailerons	ELM (own)	**0.0000**	15 min
	MLP (own)	0.0437	7 h
Machine CPU	ELM (own)	0.1116	1.5 min
	MLP (own)	**0.1015**	1 h
Servo	ELM (own)	0.1433	1.5 min
	MLP (own)	**0.0373**	30 min

For each dataset, the result obtained by the better model is presented in bold

Training times include the hyperparameter search

[1]Including the hyperparameter search time

entire training dataset. This process requires storing the hidden features matrix in memory. One could argue that it is possible to implement ELMs so that this matrix is cached on disk. However, the main advantage of extreme learning—short training times—would suffer significantly. Here we present a short theoretical study of memory requirements that one needs to train ELM on a 224×224 ImageNet dataset that consists of roughly 14 mln images. Let us presume the following assumptions:

- the training is performed on ImageNet's 1 mln images subset, used in popular challenges such as ILSVRC,
- all images are resized to constant 224×224 px size,
- the LRF-ELM architecture is configured as follows:
  - 16 convolutional filters only,
  - filter size: 5×5,
  - pooling size: 5×5,
  - pooling stride: 2.

Table 14 Hyperparameter set-up for the best found ELM models for each classification dataset

Dataset	Number of kernels	Kernel size	Pooling size	Regularisation coefficient
CIFAR-10	64	3	5	10
MNIST	64	5	7	10
NORB	48	3	3	3
GTSRB	64	7	3	3
ImageNet16	48	3	3	3
ImageNet32	8	3	3	1

Table 15 Comparison of the accuracy and training times for the best found ELM and BP solutions for given image classification datasets

Dataset	Model	Accuracy (%)	Training time[2]
MNIST	LRF-ELM (own)	98.85	1 min
	Branching and merging CNN with homogeneous filter capsules [4]	**99.79**	N/A
CIFAR-10	LRF-ELM (own)	58.63	1 min
	TResNet-XL [38]	**99.00**	Around 1 h
NORB	LRF-ELM (own)	92.61	15 s
	VB-Routing CapsuleNet [37]	**98.40**	N/A
GTSRB	LRF-ELM (own)	92.80	40 s
	Spatial transformer [1]	**99.71**	Around 1.5 h
ImageNet16	LRF-ELM (own)	4.91	1.5 min
	Wide ResNet-20, width 10 [6]	**40.06**	2.7 days
ImageNet32	LRF-ELM (own)	3.07	4 min
	Wide ResNet-28, width 10 [6]	**59.04**	13.8 days

For each dataset, the result obtained by the better model is presented in bold

Training times do not include hyperparameter search as it was not given in any referenced work

[2]Excluding the hyperparameter search time

Table 16 Qualitative comparison of ELM models and classical neural networks

Criterion	BP	ELM
# of hyperparameters	High	Low
# of trainable parameters	Very high	Lower
Training time	High	Low
Memory requirements	Low	High
Popularity	High	Low
Supplementary techniques	Many	Few
Community support	High	Low
Big gamer's interest	Very high	Low
Model's implementations	Numerous, up-to-date	Few, poorly supported

- standard 32-bit precision floating-point numbers are used.

The assumed image size and network architecture produce hidden feature maps of size $108 \times 108 \times 16$ for each image, LRF-ELM flattens such feature maps; hence, a hidden representation is a vector of 186624 elements. Therefore the hidden representation matrix $\mathbf{H}$ size is $1\text{mln} \times 186624$. Such matrix consists of 1.87×10^{11} numbers. Each of them is stored on 4 bytes, so the total size of the matrix $\mathbf{H}$ is equal to approximately 746 GB.

Using such limited number of convolutional filters leads to memory requirements that are beyond abilities of most systems. Let us assume changing convolution and pooling configuration in order to produce feature maps compressed to 32×32 size. Even in this case the matrix $\mathbf{H}$ takes more than 60 GB. Further scaling down introduces a high risk that such compressed features extracted randomly contain too little information for ELM output layer to train properly. This effect is clearly seen in the experiment results in Table 15. Using classical convolutional networks trained with backpropagation, the memory requirements are several orders of magnitude lower. There is no need to keep the entire dataset in memory; training requires loading only one mini-batch at time.

Our experience from conducting this study is described in Table 16, which shows a qualitative comparison between ELM and classical neural networks in the current state of their development.

We can sum up its content as follows:

- The number of hyperparameters is much higher in the classical neural networks, but selecting their values can be supported by the extensive literature review and help of active community members on public forums. It is easy to find heuristics on how to set hyperparameters values in the case of classical neural networks.
- The number of parameters in the classical deep networks is huge, but because the models are popular, we can use transfer learning that speeds up training. In

ELM models, the number of parameters is lower, but in challenging image data, memory requirements are so vast that it is not possible to calculate weights assuming typical hardware equipment.

- There are few publicly available ELM implementations; existing ones are usually simple and non-generic or are written in Matlab, which makes them hard to utilise in more complex use cases and popular, well-supported frameworks such as Python and PyTorch or TensorFlow.

Summing up, in this paper we referred only to the classical methods in ELMs conducting experiments on image and vector representation datasets. It should be noticed, however, that there exist some new approaches which propose promising solutions. Here, we can mention incremental learning [42], multilayer extreme learning machine (ML-ELM), and hierarchical extreme learning machine (H-ELM) described in [20], and non-iterative and fast learning algorithms for deep ELM [49]. It is a good prognostic for further development of ELM, but none of them has presented results using a challenging dataset as ImageNet nor has compared their performance to the mainstream deep learning methods.

7 Final conclusion

Based on presented research, we can sum up that the tested ELM models do not offer a real alternative compared to classical neural networks for contemporary problems characterised by complex patterns and huge datasets. The first series of experiments shows that their superiority to classical neural networks about training times is visible for networks with capacity corresponding to the time before the deep learning era. The demand for models that are trained quickly for tough image, video, or audio problems is enormous. ELM models transforming input space to high-dimensional hidden neuron space with random weights are attractive and essential when one wants to train the model quickly. But to achieve this state, we postulate to develop smart algorithms for the inverse matrix calculation, so that determining weights in the output layer for challenging datasets becomes feasible and memory efficient. There is a need to create specific mechanisms to avoid keeping the whole dataset in memory to compute weights. Although we noticed new approaches to solve these problems [20, 42, 49], contemporary demanding datasets used in classical deep neural networks are still a challenge for ELMs. Also, it seems necessary to develop generic frameworks that enable practitioners simple access to ELM models and easy development of new architectures efficiently utilising this training algorithm. Ultimately, sharing implementations should be a common practice among ELM researchers. In future works, we plan to prepare comparison with other random networks like RVFL or its extended version. There are also other promising solutions to consider—self-normalising networks that outperformed all competing methods [23].

Appendix A ELM hyperparameter sensitivity analysis

ELM are significantly less popular than classical networks. Numerous studies on training based on backpropagation allowed to establish intuitions and heuristics on how neural networks react to changes of hyperparameters. In this study, we aim to gain insight and build similar intuitions on ELMs' sensitivity to hyperparameters' values. We conducted experiments to evaluate the level of contribution that each hyperparameter has on the response variance, as defined in [39] and [52]. A theoretical model has been defined. The analysed hyperparameters are the input variables of the model. The output corresponds to the classification quality measure on a given dataset. Variance-based sensitivity analysis [41] can be performed using such model. Sensitivity indices are calculated by using decomposition of variance $\mathrm{Var}(Y)$ of model response $Y = f(X)$. In our case, the sensitivity indices were estimated based on repeated measurements within the experiments.

Sensitivity indices can be interpreted as the distribution of each parameter's impact on the variance of the model response. I-st order indices are defined for every single parameter. They indicate the share of the total model variance the variations of this single parameter have. II-nd order indices are defined for each pair of parameters. They specify how much the variance of joint changes of a pair affects the variance of the model response. To identify hyperparameter values that allow reaching high performance on multiple various datasets we utilised a pair of statistical tests used to compare classifiers [7]—Friedman's test [10] and Nemenyi's post-hoc test [31].

We performed a repeated K-fold cross-validation on every dataset. The use of cross-validation implies a lack of independence of observations in the test samples. This suggests using the dependent t-test for paired samples. However, as indicated in [3], in the case of a repeated cross-validation, the basic form of this *t*-test may lead to false and non-reproducible results. They proposed a modified paired-samples t-test, called *corrected repeated k-fold cross-validation test*. Therefore, we utilised this test to compare the average performance measures of the models.

Three ELM hyperparameters are considered:

Table 17 Sensitivity indices for the F1 measure

Dataset	σ_{F1}	I-st order indices			II-nd order indices			
		$S_{(C)}$	$S_{(h)}$	$S_{(f)}$	$S_{(C,h)}$	$S_{(C,f)}$	$S_{(h,f)}$	$S_{(C,h,f)}$
Wine	0.27	0.12	0.63	0.03	0.07	0.07	0.02	0.06
Breast cancer (W.)	0.12	0.10	0.30	0.03	0.20	0.05	0.13	0.18
Glass identification	0.16	0.16	0.42	0.03	0.24	0.07	0.03	0.05
Breast cancer (L.)	0.14	0.28	0.26	0.06	0.19	0.07	0.04	0.11
Sonar	0.21	0.35	0.26	0.06	0.11	0.10	0.02	0.10
Urban land cover	0.21	0.16	0.12	0.07	0.30	0.07	0.11	0.17
Landsat	0.21	0.18	0.25	0.06	0.30	0.07	0.06	0.09
CNAE-9	0.29	0.28	0.02	0.08	0.28	0.13	0.05	0.16
Forest cover type	0.16	0.17	0.19	0.11	0.24	0.06	0.12	0.11
2D hypercube	0.24	0.04	0.71	0.02	0.10	0.05	0.02	0.06
3D hypercube	0.40	0.12	0.52	0.04	0.15	0.08	0.03	0.06
10D hypercube	0.04	0.10	0.12	0.14	0.08	0.05	0.40	0.10
2D hyperspheres	0.25	0.11	0.58	0.04	0.08	0.09	0.05	0.05
3D hyperspheres	0.24	0.12	0.58	0.05	0.06	0.09	0.05	0.04
10D hyperspheres	0.21	0.11	0.55	0.09	0.04	0.06	0.11	0.05
100D hyperspheres	0.15	0.07	0.14	0.45	0.09	0.05	0.15	0.05
200D hyperspheres	0.12	0.05	0.11	0.51	0.02	0.06	0.21	0.05
Mean	0.20	0.15	0.34	0.11	0.15	0.07	0.09	0.09
Median	0.21	0.12	0.28	0.06	0.13	0.07	0.06	0.07

σ–overall standard deviation of results

Table 18 p-values from Friedman tests for every hyperparameter considered in the experiment

Hyperparameter	p-value
Activation function f	$2.00 \cdot 10^{-5}$
Hidden layer size coeff. h	$5.65 \cdot 10^{-11}$
Regularisation coefficient C	$3.07 \cdot 10^{-15}$

Table 19 p-value from Nemenyi test for activation function f, for F1-measure

Activation function	ReLU	Sigmoid	Tanh	Threshold
ReLU				
Sigmoid	**0.001**			
Tanh	0.900	**0.001**		
Threshold	0.079	0.191	0.191	
Average ranking	1.706	3.647	1.882	2.765

The p-values that are below the assumed significance level are expressed in bold

- Regularisation coefficient C (Eq. 12): $\{\frac{1}{10000}, \frac{3}{10000}, \frac{1}{1000}, \frac{3}{1000}, \frac{1}{100}, \frac{3}{100}, \frac{1}{10}, \frac{3}{10}, 1, 3, 10, 30, 100, 300\}$,
- Activation function f: *Sigmoid*, *Tanh*, *ReLU*, *Threshold* as it was proposed in [15],
- Hidden layer size coefficient h: $\{0.1, 0.3, 1, 3, 10, 30\}$.

Table 20 p-value from Nemenyi test for hidden layer size coefficient h, for F1-measure

Size coefficient	0.1	0.3	1.0	3.0	10.0	30.0
0.1						
0.3	0.283					
1.0	**0.001**	0.333				
3.0	**0.001**	**0.005**	0.609			
10.0	**0.001**	**0.002**	0.503	0.900		
30.0	**0.001**	**0.007**	0.662	0.900	0.900	
Average ranking	6.000	4.647	3.353	2.353	2.235	2.412

The p-values that are below the assumed significance level are expressed in bold

In the experiment, 17 datasets for classification task were used. The values of sensitivity indices are shown in Table 17. Table 18 shows the p-values from Friedman tests. All of them are below the assumed significance level of $\alpha = 0.05$, so it can be stated that the rankings for different values of hyperparameters on different datasets differ significantly. For this reason, Nemenyi tests were conducted in order to evaluate the pairs of hyperparameter values yielding particularly differing results. The results are presented in Tables 19, 20, and 21. The p-values that are below the assumed significance level are expressed in bold. In addition, the tables present the average ranking values that are useful for the derivation of conclusions.

Table 21 p-values from Nemenyi test for regularisation coefficient C, for F1-measure

Regularisation coefficient	0.0001	0.0003	0.001	0.003	0.01	0.03	0.1	0.3	1	3	10	30	100	300
0.0001														
0.0003	0.900													
0.001	0.325	0.900												
0.003	**0.022**	0.497	0.900											
0.01	**0.002**	0.140	0.900	0.900										
0.03	**0.001**	**0.016**	0.550	0.900	0.900									
0.1	**0.001**	**0.014**	0.523	0.900	0.900	0.900								
0.3	**0.001**	0.071	0.838	0.900	0.900	0.900	0.900							
1	**0.019**	0.470	0.900	0.900	0.900	0.900	0.900	0.900						
3	0.382	0.900	0.900	0.900	0.900	0.497	0.470	0.785	0.900					
10	0.838	0.900	0.900	0.890	0.497	0.114	0.101	0.325	0.864	0.900				
30	0.900	0.900	0.890	0.252	**0.049**	**0.004**	**0.003**	**0.022**	0.231	0.900	0.900			
100	0.900	0.900	0.353	**0.025**	**0.002**	**0.001**	**0.001**	**0.001**	**0.022**	0.411	0.864	0.900		
300	0.900	0.707	**0.038**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.049**	0.300	0.900	0.900	
Average ranking	11.000	9.235	7.235	5.824	4.941	3.941	3.882	4.588	5.765	7.353	8.353	9.765	10.941	12.176

The p-values that are below the assumed significance level are expressed in bold

Analysing Table 17, we can observe that among the selected hyperparameters, variations in the size of the hidden layer have the greatest impact on the variance of the model performance. This phenomenon is demonstrated by the values of the $S_{(h)}$ index.

It can be noted that the values of the $S_{(f)}$ index (Table 17) increase significantly as the dimensionality of the hypersphere classification problem increases. The same can be observed for hypercube vertices classification. Simultaneously, the values of the $S_{(h,f)}$ index increase, which means that the importance of properly choosing the activation function increases when increasing the problem dimensionality.

The changes of the C and f hyperparameters demonstrate a comparable contribution to the overall variance considering the first order interaction. Among the second order indices, the index $S_{(C,f)}$ is usually the one with the lowest value. Thus, the optimisation of the regularisation coefficient and the activation function can be carried out independently.

Nemenyi test for activation function f revealed the existence of two pairs of activation functions, which differ significantly with regard to the results achieved on various datasets. Both pairs include the sigmoid function. Considering the average ranking positions (approximately 3.5 for the sigmoid and about 2 for the others), it can be concluded that the sigmoid function contributes least to the improvement of the classification results.

Nemenyi test revealed significant differences between the smallest two values of h (0.1 and 0.3) and the four largest values (1, 3, 10, and 30). Just like in the case of the activation function, it can be concluded that small values of the h hyperparameter have a little positive effect on the performance quality. There are no indications to exclude $h = 0.1$ and $h = 0.3$ from the process of hyperparameter tuning.

Based on the Nemenyi test for the regularisation coefficient C (Table 21), it is easy to observe that even the lowest values of the average ranking position (approximately 4) are noticeably higher than the ranking values of the other hyperparameters (approximately 2). This may indicate a high sensitivity to this hyperparameter and a lack of values that are clearly better for multiple datasets. The best ranking positions were obtained for C values in the range from 0.01 to 0.3.

Appendix B Artificial datasets for classification

In the experiments we have used three types of artificial classification datasets:

Nested hyperspheres Dataset consists of points lying on two concentric hyperspheres with different radii, which represent two classes. To generate random points uniformly distributed on a hypersphere we used the procedure shown in [36]. We added a Gaussian noise to all points.

Hypercube vertices This is a generalisation of the XOR classification task. In this dataset, each hypercube vertex is assigned one of two labels in such a way that the classes are not linearly separable. Knowing that a linear SVM achieves 100% accuracy for linearly separable classes, we repeat

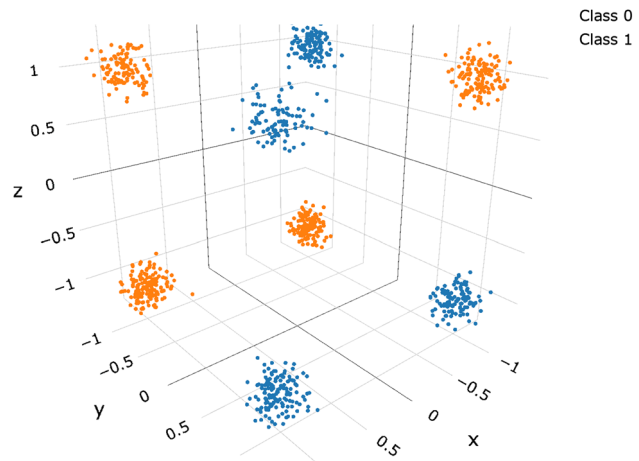


Fig. 6 Visualisation of the hypercube vertices binary classification dataset

random vertex-class assignments until the SVM is unable to fit the data. Once a linearly non-separable labelling is obtained, the vertices are sampled to the dataset and Gaussian noise is added. Figure 6 shows a visualisation of the dataset generated in 3-D space.

Intertwined spirals This is a binary classification task. Samples of a single class lie on a 2-D Archimedean spiral or on a 3-D conical spiral. Data points of the other class lie on a similar rotated spiral. 2-D spirals were generated using formulas in Eqs. 17 and 18 defined in polar coordinates. We converted points to Cartesian coordinates and added Gaussian noise. We used the following values of coefficients: $a_1 = a_2 = 1$ and $b_1 = b_2 = 0.5$. Figure 7 presents visualisations of such datasets.

$$r = a_1 \cdot \theta + b_1 \tag{17}$$

$$r = -a_2 \cdot \theta + b_2 \tag{18}$$

where (θ, r) are the polar coordinates (angle and radius) and a, b are parameters of Archimedean spiral.

3-D spirals were generated according to the Eqs. 19 and 20 in the cylindrical space. The following values of parameters were used in this study. $a_1 = a_2 = 1$, $b_1 = 0.5$, $c_1 = 1.2$, $b_2 = 3.75 \cdot b_1$, $c_2 = 3 \cdot c_1$.

$$\begin{cases} r = a_1 \cdot \theta + b_1 \\ h = c_1 \cdot \theta \end{cases} \tag{19}$$

$$\begin{cases} r = a_2 \cdot \theta + b_2 \\ h = c_2 \cdot \theta \end{cases} \tag{20}$$

where (θ, r, h) are the cylindrical coordinates (angle, radius, and height) and a, b, c are the parameters of the spiral.

Regression datasets are generated similarly. They are designed to embed into the data important properties of real-valued functions: periodicity and trend.

Linear function $y = \mathbf{a}^\top \mathbf{x} + b$.: It is the simplest approach to trend modelling. It is possible to generate a set in a space of any dimensionality. The values of the slope $\mathbf{a}$ and the bias b are drawn randomly from the $[-5, 5]$ range.

Second degree polynomial of many variables.: Coefficients and the bias are randomly drawn from the $[-5, 5]$ range. There may be correlations between variables, which must be correctly reproduced by the model.

Sine, cosine: Periodicity modelling. Data generated on the $[-2\pi, 2\pi]$ domain.

SinC: Function expressed by Eq. 21. It is a mixture of periodicity and trend.

$$y = \begin{cases} \dfrac{\sin(x)}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{21}$$

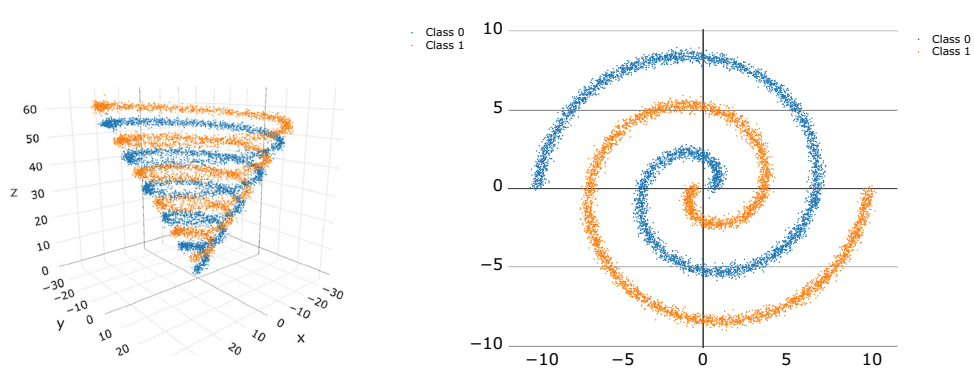


Fig. 7 Visualisation of the intertwined spirals datasets in 2D and 3D spaces

Friedman splines I, II, and III: The splines were proposed as reference functions for the study of performance of regression models. The first one is a synthetic curve, designed to test the ability to detect relationships between variables. The remaining two correspond to physical phenomena. The splines are defined by Eqs. 22, 23, and 24 . ϵ denotes a random noise. Spline I has an interesting characteristic—it is defined using only 5 variables, however the dataset consists of more variables, so the function values depend only on some of the features.

$$f_1(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - \tfrac{1}{2})^2 + 10x_4 + 5x_5 + \epsilon \tag{22}$$

$$f_2(x) = (x_1^2 + (x_2 x_3 - (\tfrac{1}{x_2 x_4})^2)^{\frac{1}{2}}) + \epsilon \tag{23}$$

$$f_3(x) = \arctan\left(\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1}\right) + \epsilon \tag{24}$$

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Arcos-García Á, Álvarez-García JA, Soria-Morillo LM (2018) Deep neural network for traffic sign recognition systems: an analysis of spatial transformers and stochastic optimisation methods. Neural Netw 99:158–165
2. Bottou L (2012) Stochastic gradient descent tricks. In: Montavon G, Orr GB, Müller KR (eds) Neural networks: tricks of the trade, 2nd edn. Springer, Berlin, pp 421–436. https://doi.org/10.1007/978-3-642-35289-8_25
3. Bouckaert R, Frank E (2004) Evaluating the replicability of significance tests for comparing learning algorithms. In: Dai H, Srikant R, Zhang C (eds) Pacific Asia knowledge discovery and data mining. Springer, Berlin, pp 3–12
4. Byerly A, Kalganova T, Dear I (2020) A branching and merging convolutional network with homogeneous filter capsules. arXiv:2001.09136
5. Cao W, Wang X, Ming Z, Gao J (2018) A review on neural networks with random weights. Neurocomputing 275:278–287. https://doi.org/10.1016/j.neucom.2017.08.040
6. Chrabaszcz P, Loshchilov I, Hutter F (2017) A downsampled variant of ImageNet as an alternative to the CIFAR datasets. arXiv:1707.08819
7. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
8. Dua D, Graff C (2017) UCI Machine learning repository. http://archive.ics.uci.edu/ml
9. Dudek G (2020) Are direct links necessary in random vector functional link networks for regression? In: Rutkowski L, Scherer R, Korytkowski M, Pedrycz W, Tadeusiewicz R, Zurada JM (eds) Artificial intelligence and soft computing. Springer, Cham, pp 60–70
10. Friedman M (1940) A comparison of alternative tests of significance for the problem of *m* rankings. Ann Math Stat 11(1):86–92
11. Ganaie MA, Tanveer M, Suganthan PN (2020) Minimum variance embedded random vector functional link network. In: Yang H, Pasupa K, Leung AC, Kwok JT, Chan JH, King I (eds) Neural information processing - 27th international conference, ICONIP 2020, Bangkok, Thailand, November 18-22, 2020, Proceedings, Part V, Springer, Communications in Computer and Information Science, vol 1333, pp 412–419, https://doi.org/10.1007/978-3-030-63823-8_48,
12. Huang G, Huang GB, Song S, You K (2015a) Trends in extreme learning machines: a review. Neural Netw 61:32–48
13. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE International joint conference on neural networks, IEEE, vol 2, pp. 985–990
14. Huang GB, Liang N, Rong HJ, Saratchandran P, Sundararajan N (2005) On-line sequential extreme learning machine. In: Proceedings of the IASTED international conference on computational intelligence, vol 2005, pp. 232–237
15. Huang GB, Zhu QY, Mao KZ, Siew CK, Saratchandran P, Sundararajan N (2006a) Can threshold networks be trained directly? IEEE Trans Circuits Syst II: Express Briefs 53(3):187–191
16. Huang GB, Zhu QY, Siew CK (2006b) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501
17. Huang GB, Bai Z, Kasun LLC, Vong CM (2015b) Local receptive fields based extreme learning machine. IEEE Comput Intell Mag 10(2):18–29
18. Huang Z, Yu Y, Gu J, Liu H (2017) An efficient method for traffic sign recognition based on extreme learning machine. IEEE Trans Cybern 47(4):920–933
19. Jarrett K, Kavukcuoglu K, Ranzato A, LeCun Y, Ranzato M, LeCun Y (2009) What is the best multi-stage architecture for object recognition? In: Proceedings of the IEEE international conference on computer vision, IEEE, pp. 2146–2153
20. Jiang XW, Yan TH, Zhu JJ, He B, Li WH, Du HP, Sun SS (2020) Densely connected deep extreme learning machine algorithm. Cognit Comput 12:1866–9964
21. Kasun L, Zhou H, Huang G, Vong C (2013) Representational learning with extreme learning machine for big data. IEEE Intell Syst 28:31–34

22. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio Y, LeCun Y (eds) 3rd International Conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, arXiv:1412.6980
23. Klambauer G, Unterthiner T, Mayr A, Hochreiter S (2017) Self-normalizing neural networks. In: Proceedings of the 31st international conference on neural information processing systems, Curran Associates Inc., NIPS'17, pp. 972–981
24. Krizhevsky A, Sutskever I, Hinton GE (2012). ImageNet classification with deep convolutional neural networks, vol NIPS'12. Curran Associates Inc., USA, pp. 1097–1105
25. Lai J, Wang X, Li R, Lei YSL (2020) BD-ELM: A regularized extreme learning machine using biased dropconnect and biased dropout. Math Probl Eng 2020:7
26. Linnainmaa S (1976) Taylor expansion of the accumulated rounding error. BIT Numer Math 16(2):146–160
27. MacMichael D, Si D (2018) Machine learning classification of tree cover type and application to forest management. Int J Multimed Data Eng Manage 9(1):1–21
28. Mangasarian OL, Street WN, Wolberg WH (1995) Breast cancer diagnosis and prognosis via linear programming. Oper Res 43(4):570–577
29. Meyer M (1989) StatLib - Datasets archive. http://lib.stat.cmu.edu/datasets/
30. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning, Omnipress, pp. 807–814
31. Nemenyi PB (1963) Distribution-free multiple comparisons. Princeton University, Princeton (**PhD thesis**)
32. Pao YH, Park GH, Sobajic DJ (1992) Neural-net computing and the intelligent control of systems. Int J Control 56(2):263–289
33. Pao YH, Park GH, Sobajic DJ (1994) Learning and generalization characteristics of the random vector functional-link net. Neurocomputing 6(2):163–180
34. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d' Alche-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems 32. Curran Associates Inc, New York, pp 8024–8035
35. Pieterse J, Mocanu DC (2019) Evolving and understanding sparse deep neural networks using cosine similarity. arXiv:1903.07138
36. Poland J (2000) Three different algorithms for generating uniformly distributed random points on the N-sphere, http://www-alg.ist.hokudai.ac.jp/~jan/randsphere.pdf
37. Ribeiro F, Leontidis G, Kollias S (2020) Capsule routing via variational Bayes. https://doi.org/10.1609/aaai.v34i04.5785, arXiv:1905.11455
38. Ridnik T, Lawen H, Noy A, Friedman I, Baruch EB, Sharir G (2021) TResNet: High performance GPU-dedicated architecture. 2021 IEEE Winter conference on applications of computer vision (WACV) pp. 1399–1408, arXiv:2003.13630
39. Saltelli A (2001) Sensitivity analysis for importance assessment. In: Proceedings of the 3rd international symposium on sensitivity analysis of model output pp. 3–18
40. Schrauwen B, Verstraeten D, Campenhout JV (2007) An overview of reservoir computing: theory, applications and implementations. In: Proceedings of the 15th european symposium on artificial neural networks, pp. 471–482
41. Sobol IM, Kucherenko SS (2005) Global sensitivity indices for nonlinear mathematical models, review. Wilmott 1:56–61
42. Song S, Wang M, Lin Y (2020) An improved algorithm for incremental extreme learning machine. Syst Sci Control Eng 8(1):308–317
43. Suganthan PN (2018) Letter: on non-iterative learning algorithms with closed-form solution. Appl Soft Comput 70:1078–1082
44. Swiecicki M (2009). An algorithm based on the construction of Braun's cathode ray tube as a novel technique for data classification. In: Lecture notes in computer science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, Berlin, Heidelberg, vol 5864 LNCS, pp. 710–719
45. Tao X, Zhou X, Lin He Y, Aamir Raza Ashfaq R (2016) Impact of variances of random weights and biases on extreme learning machine. J Softw 11(5):440–454
46. Torgo L (2012) Regression data sets – Luis Torgo Repository. http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html
47. Vuković N, Petrović M, Miljković Z (2018) A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression. Appl Soft Comput 70:1083–1096
48. Zarndt F (1995) A comprehensive case study: an examination of machine learning and connectionist algorithms. Master's thesis, Brigham Young University. Department of Computer Science
49. Zhang J, Li Y, Xiao W, Zhang Z (2020) Non-iterative and fast deep learning: multilayer extreme learning machines. J Frankl Inst 357(13):8925–8955
50. Zhang LZ, Suganthan P (2016) A comprehensive evaluation of random vector functional link networks. Inf Sci 367–368:1094–1105
51. Zhou H, Huang G, Lin Z, Wang H, Soh YC (2015) Stacked extreme learning machines. IEEE Trans Cybern 45(9):2013–2025
52. Zhou X, Lin H, Lin H (2008) Global sensitivity analysis. Springer, Boston, pp 408–409
53. Zong W, Huang GB, Chen Y (2013) Weighted extreme learning machine for imbalance learning. Neurocomputing 101:229–242