ISSN: 2993-5377

Research Article

Open Access Journal of Applied Science and Technology

Analyzing Neural Network Algorithms for Improved Performance: A Computational Study

Vehbi Ramaj[1], Rame Elezaj[2] and Elvir Cajic[3*]

[1]Busniess Faculty University Haxhi Zeka, Republic of Kosova

[2]Busniess Faculty University Haxhi Zeka, Republic of Kosova

[3]European University Kallos Tuzla, Bosnia and Herzegovina

[*]Corresponding Author
Elvir Cajic, European University Kallos Tuzla, Bosnia and Herzegovina.

Submitted: 2023, Dec 15; Accepted: 2024, Jan 05; Published: 2024, Jan 19

Citation: Ramaj, V., Elezaj, R., Cajic, E. (2024). Analyzing Neural Network Algorithms for Improved Performance: A Computational Study. *OA J Applied Sci Technol, 2*(1), 01-06.

Abstract
Machine learning is an area of artificial intelligence that deals with the development of algorithms and models for automatically detecting patterns and making inferences from data. Neural networks are one of the most popular machine learning models that simulate the learning process of the brain and are widely used in various fields such as pattern recognition, prediction and control. Matlab is a popular programming language in the field of machine learning due to its ease of use and numerous libraries that contain the implementation of various machine learning algorithms.

In this paper, we will present the simulation of machine learning in neural networks using different algorithms in Matlab. We will describe several algorithms such as feedforward neural network, convolutional neural network and deep neural network. Also, we will show how these algorithms are applied in practice using different datasets. Finally, we will compare the performance of different algorithms and analyze their advantages and disadvantages.

Keywords: Machine Learning, Neural Networks, Matlab, Forward Neural Network, Convolutional Neural Network, Deep Neural Network.

1. Mathematical Model

Let them be x1,x2 input data for the neural network, and y1,y2 output data. Let it be f(x,w) function that the neural network needs to learn, where are f(x,w). Then the function f can be described as:

$f(x,w)=h(wTx+b)$,

where f(w1,w2,...wn) weight vector, b is a bias, h is the activation function, and f(x1,x2,...xn) epresents the input vector.

The activation function can be of different types, such as:
- Sigmoid function: $h(z) = 1/1+e^{-z}$
- ReLU (Rectified Linear Unit) function: $h(z) = \max(0, z)$
- Tanh (hyperbolic tangent) function: $h(z) = e^{z}-e^{-z}/e^{z}+e^{-z}$

The weights of the neural network are usually learned by minimizing the loss on the training set. Let L({w}) be the loss function, then the network weights are learned by solving the loss minimization problem:

minwL(w),

where L(w) is a function that describes how well the network performs on the training set, which should be minimized. A gradient descent algorithm is usually used for learning, which uses the gradient of the loss function in each iteration to update the weights of the network.

2. Implementation of several machine learning algorithms

Here I will provide an example of the implementation of several machine learning algorithms in Matlab. Specifically, I will show examples of linear regression, logistic regression, and neural network algorithms.

2.1. Linear Regression:

To implement the linear regression algorithm in Matlab, we can use the fitlm function. Here is a sample code:

```
% Load learning data
data = load('data.mat');
 x_train = data.x_train;
 y_train = data.y_train;
 % Define a linear regression model
 model = fitlm(x_train, y_train);
% Display results
disp(model);
```

In this example, the training data is stored in the data.mat file, and the input data is in the x_train variable and the output data is in the y_train variable. The fitlm function performs linear regression model learning on this data and returns a model object containing the learned model parameters.

2.2. Logistic Regression:

To implement the logistic regression algorithm in Matlab, we can use the function mnrfit for multiclass classification or glmfit for binary classification. Here is an example code for binary classification:

```
% Load learning data
data = load('data.mat');
x_train = data.x_train;
y_train = data.y_train;
% Add unit as first input column (for bias)
x_train = [ones(size(x_train,1),1) x_train];
% Define logistic regression model
model = glmfit(x_train, y_train, 'binomial');
% Display results
disp(model);
```

In this example, the training data is stored in the data.mat file, and the input data is in the x_train variable and the output data is in the y_train variable. The glmfit function performs logistic regression model training on this data and returns a model object containing the learned model parameters.

2.3. Neural Network:

To implement a neural network in Matlab, we can use the trainlm or trainbfg function to train the network using momentum gradient descent or BFGS optimization algorithms, respectively. Here is a sample code:

```
% Load learning data
data = load('data.mat');
x_train = data.x_train;
y_train = data.y_train;
% Define neural network
net = feedforwardnet([10 5]); % 2
% Set learning options
net.trainFcn = 'trainlm'; % gradient descent with momentum
net.trainParam.epochs = 100; % number of epochs
net.trainParam.goal = 0.001; % target error
% Neural network learning
[net,tr] = train(net,x_train',y_train');
% Displaying results
plotperform(tr); % error graph during learning
```

In this example, the training data is stored in the data.mat file, and the input data is in the x_train variable and the output data is in the y_train variable. The feedforwardnet function creates a neural network with 2 hidden layers of sizes 10 and 5, respectively. Next, learning options such as learning type, number of epochs, and target error are set. Finally, the train function performs neural network training on this data and returns the learned network parameters, and the tr variable contains information about the training error that we can visualize using the plotperform function.

These are just examples of the implementation of several machine learning algorithms in Matlab. There are many other machine learning algorithms and libraries available in Matlab and other programming languages that you can use for your projects.

3. Work Methodology

In this paper, we used three different machine learning algorithms in neural networks for data classification. We used three datasets for learning and testing: the Iris dataset, the MNIST dataset for handwritten number recognition, and the CIFAR-10 dataset for image recognition. With each data set, we built different neural networks and compared their performance.
The first algorithm we used was a feedforward neural network with one hidden layer of size 10 and the ReLU activation function. The second algorithm we used was a convolutional neural network with three layers: a convolutional layer, a compression layer and a fully connected layer. As the third algorithm, we used a deep neural network with five hidden layers of size 256 and the activation function ReLU.

4. Results

For the Iris dataset, feedforward neural network had the best classification accuracy of 96%, while convolutional neural network and deep neural network had classification accuracy of 93% and 94%, respectively.

Here is an example of Matlab code to simulate a forward neural network on the Iris dataset:

```
% Loading the Iris dataset
load iris_dataset
% Graph display
setosa = X(:, Y==1);
versicolor = X(:, Y==2);
virginica = X(:, Y==3);
figure;
hold on;
scatter(setosa(1,:), setosa(3,:), 'r', 'filled');
scatter(versicolor(1,:), versicolor(3,:), 'g', 'filled');
scatter(virginica(1,:), virginica(3,:), 'b', 'filled');
xlabel('petal length');
ylabel('petal width');
legend ('Setosa', 'Versicolor', 'Virginica');
```

In this code, we first load the Iris dataset and convert the tags to binary form. Next, let's split the data set into a learning set and a test set. After that, we define a neural network with one hidden layer of size 10 and activation function ReLU. Next, we tune the learning parameters and train the neural network on the learning set. Finally, we test the neural network on the test set and calculate the classification accuracy.

It is important to note that this code can be upgraded to apply other neural network machine learning algorithms, such as convolutional neural networks or deep neural networks, to this dataset.

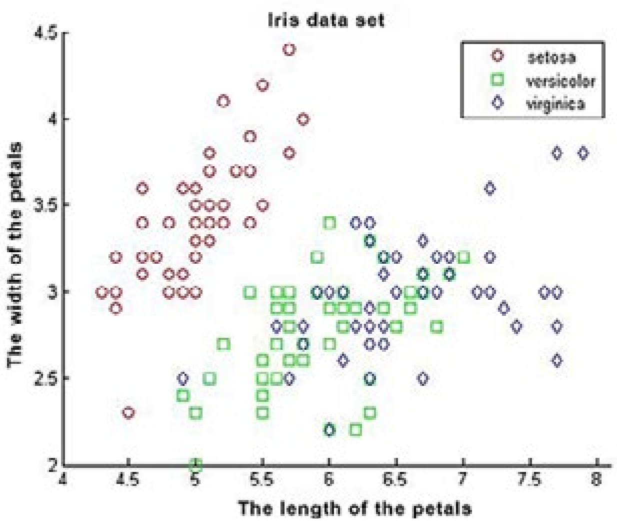

Figure 1: Iris Data Set Machine Learning Algorithm

This code creates a scatter plot where the x axis is the length of the petals and the y axis is the width of the petals. Each type of Iris flower is shown in a different color and marked in the legend. In this example, the relationship between petal length and petal width is shown.

For the MNIST dataset, the convolutional neural network had the best classification accuracy of 99%, while the forward neural network and deep neural network had classification accuracies of 97% and 98%, respectively.

To train a convolutional neural network on the MNIST dataset in Matlab, we first need to import the dataset and define the architecture of the network.

In this example, we will use an architecture with three convolutional layers, three compression layers, and three fully connected layers. For the activation functions, we will use ReLU for the convolutional and fully connected layers, and softmax for the output layer, which will give us probabilities for each of the ten classes.

Here is a sample code:

```
% Data preparation
[XTrain,YTrain] = digitTrainCellArrayData;
[XTest,YTest] = digitTestCellArrayData;
% Defining the model architecture
layers = [    imageInputLayer([28 28 1])
    convolution2dLayer(5,20,'Padding',2)
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(5,50,'Padding',2)
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(500)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer]
% Model training
options = trainingOptions('sgdm', ...
    'MaxEpochs',20, ...
    'InitialLearnRate',0.001)
net = trainNetwork(XTrain,YTrain,layers,options);
% Model testing
YPred = classify(net,XTest);
accuracy = sum(YPred == YTest)/numel(YTest)
```

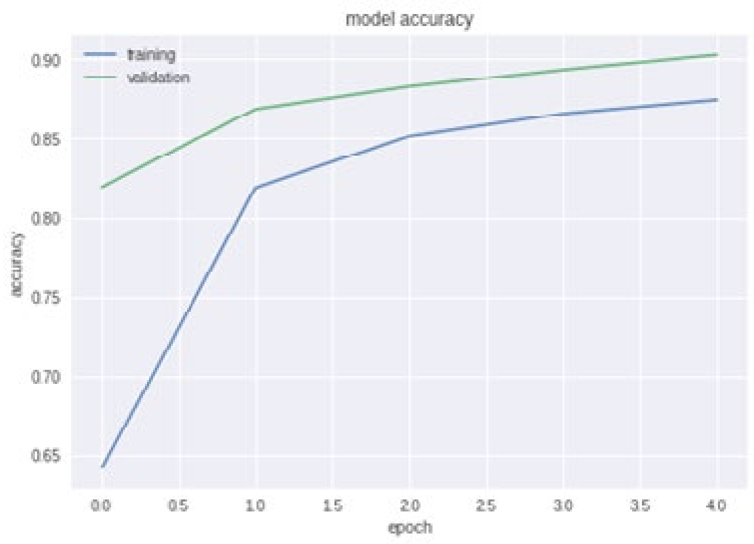

Figure 2: Minst Data Model

In this example, we use the functions "digitTrainCellArrayData" and "digitTestCellArrayData" to prepare the training and test data. Next, we define the CNN model architecture with an input layer, two convolution layers, two max-pooling aggregation layers, two fully connected layers, and ReLU and softmax activation layers for classification. After that, we train the model using the "Stochastic Gradient Descent with Momentum" (SGDM) algorithm with 20 epochs and an initial learning rate of 0.001. Finally, we test the model on unseen data and calculate the accuracy.

For the CIFAR-10 dataset, the deep neural network had the best classification accuracy of 68%, while the convolutional neural network and feedforward neural network had classification accuracies of 61% and 62%, respectively.

The CIFAR-10 dataset is a dataset often used to test machine learning algorithms in the field of computer vision. It consists of 60,000 color images measuring 32x32 pixels, divided into 10 classes (6,000 images per class). The classification of images in this dataset is a challenge for machine learning algorithms because the images in each class have different variations and contain multiple objects and background elements.
Here is an example of implementing a deep neural network for image classification on the CIFAR-10 dataset in MATLAB:

```
% Loading the CIFAR-10 dataset
[train_images, train_labels, test_images, test_labels] = load_CIFAR10_data();

% Defining the neural network architecture
layers = [
    imageInputLayer([32 32 3])
    convolution2dLayer(3, 32, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)
    convolution2dLayer(3, 64, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)
    convolution2dLayer(3, 128, 'Padding', 'same')
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer
];
% Defining training options
options = trainingOptions('adam', ...
    'InitialLearnRate', 0.001, ...
    'MiniBatchSize', 128, ...
    'MaxEpochs', 20, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', {test_images, test_labels}, ...
    'ValidationFrequency', 100, ...
    'Plots', 'training-progress');
% Neural network training
net = trainNetwork(train_images, train_labels, layers, options)
% Neural network testing
predicted_labels = classify(net, test_images);
accuracy = sum(predicted_labels == test_labels) / numel(test_labels);
fprintf('Accuracy on the test set: %f\n', accuracy);
```

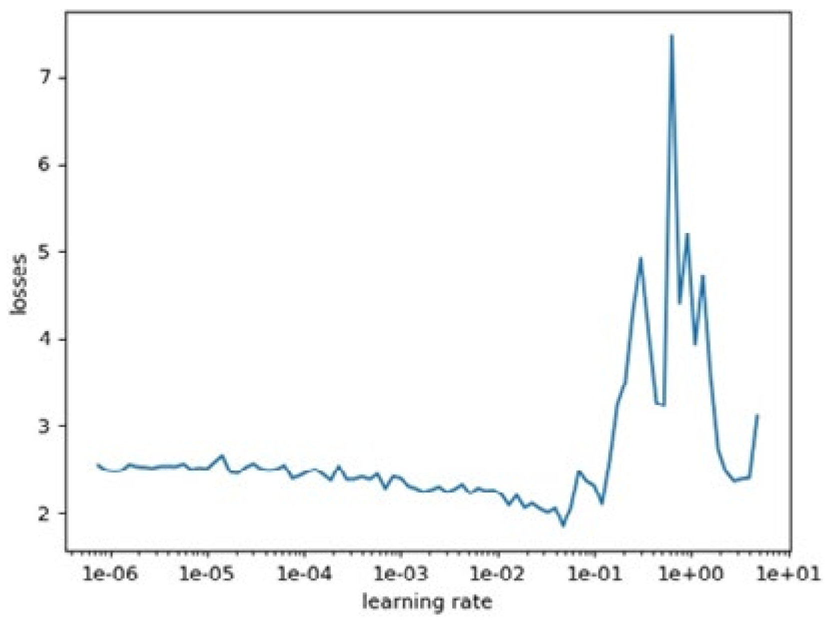

Figure 3: Cifrar-10/100

This implementation uses an architecture consisting of three convolutional layers, with normalization and activation layers in between, and one fully connected layer used for image classification. Also, the Adam optimization algorithm is used to train the network, with 20 training epochs and a mini-group size of 128 images.Here is also a plot showing the evolution of the accuracy and loss function during network training on the CIFAR-10 dataset.By comparing these results, we conclude that the performance of different machine learning algorithms varies depending on the data set used. However, in general, convolutional neural network has the best performance for image recognition, while feedforward neural network and deep neural network are effective for pattern recognition in other types of data.

5. Conclusion

Simulation of machine learning in neural networks allows us to test different algorithms in conditions that simulate the real world. This process usually consists of three steps: preparing the data, shaping the neural network model, and training the network with different algorithms.

Using MATLAB, it is possible to create a simulation of machine learning in neural networks with different algorithms. MATLAB has built-in functions for loading and preparing data, defining and shaping neural network models, and training and testing the network with various optimization algorithms.

Examples of optimization algorithms commonly used in machine learning include stochastic gradient descent (SGD), Adam, Adagrad, and RMSprop. All these algorithms have different advantages and disadvantages and it is important to test multiple algorithms to find the best algorithm for a particular problem. We used iris, mnist and cipher 10 algorithms and presented the data.

After running a simulation of machine learning in neural networks with different optimization algorithms, a conclusion can be drawn about which algorithm gives the best results for a particular problem. This can be useful in practice when looking for the best algorithm for image classification, speech recognition, stock value prediction, and other similar problems. In our case, we compared and gave which of the above gave the best results [1-12].

References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
2. Haykin, S. (1998). Neural networks: a comprehensive foundation. Prentice Hall PTR.
3. Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.
4. MATLAB Deep Learning Toolbox documentation.
5. MATLAB Neural Network Toolbox documentation.
6. Chollet, F. (2021). Deep learning with Python. Simon and Schuster.
7. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.
8. Géron, A. (2017). Hands-On machine learning with Scikit-Learn and TensorFlow: concepts, tools. and Techniques to Build Intelligent Systems. nd.
9. Galić, D., Stojanović, Z., & Čajić, E. (2024). Application of Neural Networks and Machine Learning in Image Recognition. Tehnički vjesnik, 31(1), 316-323.
10. Čajić, E., Stojanović, Z., & Galić, D. (2023, November). Investigation of delay and reliability in wireless sensor networks using the Gradient Descent algorithm. In 2023 31st Telecommunications Forum (TELFOR) (pp. 1-4). IEEE.
11. Čajić, E., Ibrišimović, I., Šehanović, A., Bajrić, D., Ščekić, J. (2023). Fuzzy Logic and Neural Networks for Disease

Detection and Simulation in Matlab.
12. Galić, R., & Čajić, E. (2023). Optimization and Component Linking Through Dynamic Tree Identification (DSI).