ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20230621000751
Preliminary communication

# Application of Neural Networks and Machine Learning in Image Recognition

Dario GALIĆ*, Zvezdan STOJANOVIĆ, Elvir ČAJIĆ

**Abstract:** Artificial neural networks find extensive applications in various fields, including complex robotics, computer vision, and classification tasks. They are designed to mimic the highly complex, nonlinear, and parallel computational abilities of the human brain. Just like neurons in the brain, artificial neural networks can be organized to perform rapid and specific computations, such as perception and motor control. Drawing insights from the behavior of biological neural networks and their learning and adaptive capabilities, these technical counterparts have been developed to simulate the properties of biological systems. This paper concentrates on two main areas. Firstly, it explores the approximation of image recognition for healthy individuals using artificial neural networks. Secondly, it investigates the identification of kidney conditions associated with common kidney diseases that affect the global population. Specifically, the paper examines polycystic kidney disease, kidney cysts, and kidney cancer. The ultimate goal is to utilize machine learning algorithms to aid in diagnosing kidney diseases by analyzing various samples.

**Keywords**: image recognition; medical diagnosis; neural networks

## 1 INTRODUCTION

Neural networks are closely inspired by biological processes involved in information processing, particularly those observed in the nervous system, where the fundamental unit is the neuron or nerve cell (as depicted in Fig. 1). The neuron serves as the basic functional component of nervous tissue, including the brain. It comprises a cell body, also known as the soma, which houses the cell nucleus. Extending from the cell body are numerous fibers called dendrites, forming an intricate network-like structure around the cell, along with a single, elongated fiber known as the axon. The axon can extend over considerable distances, typically up to one centimeter, and in extreme cases, it may even reach one meter. Furthermore, the axon branches into structures and substructures that establish connections with the dendrites and cell bodies of other neurons. These interconnected junctions between neurons are referred to as synapses [1]. Each neuron forms synapses with other neurons, and the number of such synapses can vary from a few tens to several hundred thousand. Generally, when a neuron is at rest, it receives signals in the form of electrochemical impulses transmitted through the dendrites from other neurons.

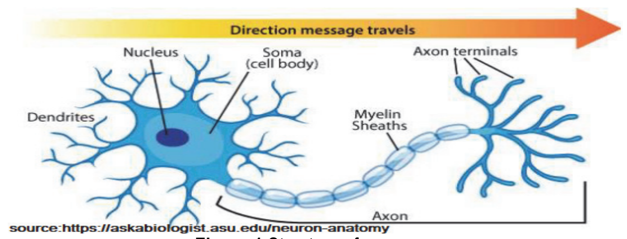

Direction message travels
Nucleus
Soma (cell body)
Axon terminals
Dendrites
Myelin Sheaths
Axon
source:https://askabiologist.asu.edu/neuron-anatomy

**Figure 1** Structure of neurons

### 1.1 Mathematical Model of Neurons

Neural networks, also known as Artificial Neural Networks (ANN), are algorithmic systems inspired by the human brain's functioning. When we open our eyes, the data we perceive is processed by neurons in our brain, which recognize the surrounding environment. Similarly, neural networks work by processing large sets of data and extracting patterns from them. Unlike the natural neurons in our brains, artificial neural networks are designed to simulate biological brains and solve specific problems. They consist of interconnected processing elements (neurons) that learn from examples during the training process. This training configures the ANN for specific tasks, such as pattern recognition, data classification, image recognition, or speech recognition. Neural networks are particularly useful for modeling highly non-linear systems with complex relationships among variables. The advantages of neural networks include adaptive learning, where they learn from training data or initial experience, and self-organization, where an ANN can create its representation of the information it receives during learning. The structure of the article is as follows:
Chapter 2 provides the mathematical basis necessary for understanding the operation of neural networks.
Chapter 3, the learning algorithm is described, with a focus on the possibility of using artificial neural networks in the detection of kidney cancer through the analysis of images of diseased and healthy kidneys.
Chapter 4 is dedicated to testing and experimental results, providing a comprehensive explanation of the research findings and identifying any limitations in this approach.
Chapter 5 describes the process of preparing images for further processing in the process of image recognition.

At the end of the article, there is a discussion and a conclusion about what was done in the article.

By employing neural networks, researchers can take advantage of their ability to extract meaning from complex or imprecise data, enabling the discovery of patterns and trends that might be too intricate for humans or other computer techniques to discern.

## 2 RELATED WORK

ANNs have shown significant potential in the field of medical diagnosis, offering advantages in early and accurate disease detection. They can assist in improving prediction accuracy, enabling timely treatment and early detection, especially in cases of chronic diseases like oral cancer and chronic kidney disease (CKD), [2]. In [3], an overview of ANN applications in medical diagnostics is provided, highlighting the promising performance advantages in disease diagnosis. Paper [4] demonstrates that using ANN for oral cancer risk prediction improves prediction accuracy, which is crucial for early detection and effective treatment. CKD is a prevalent global health issue affecting approximately 10% of the world's population [5]. Early detection of CKD is crucial for timely treatment to slow down the progression of the disease. Machine learning models, including ANN, have demonstrated fast and accurate recognition performance, making them effective tools for aiding clinicians in early CKD prediction and diagnosis [5]. Paper [6] introduces the Adaptive Hybridized Deep Convolutional Neural Network (AHDCNN) for early prediction and diagnosis of CKD, emphasizing the significance of machine learning techniques in medical diagnosis. In [7], a novel approach for predicting chronic kidney disease is presented, utilizing data mining and machine learning techniques to achieve accurate classification with high rates of precision.

The architecture of an ANN typically consists of three groups, or layers, of units (Fig. 2): the input layer, hidden layer, and output layer [3, 4].

Input Units: Represent raw information that is fed into the network, carrying data to be processed.

Hidden Units: These units' activities are determined by the input units' activities and the weights of the connections between the input and hidden units.

Output Units: The behavior of the output units depends on the activities of the hidden units and the weights between the hidden and output units, generating the final output or prediction.

This layer structure allows ANNs to process data through interconnected units, extracting patterns and relationships to make accurate predictions or classifications in medical diagnosis tasks. In the diagram, the circles represent neurons. The first layer is the input layer, while the last layer is the output layer, and the layers in between are the hidden layers. The connections between neurons are called synapses. Neurons and synapses are the main components of an Artificial Neural Network (ANN) and are responsible for conducting mathematical operations.

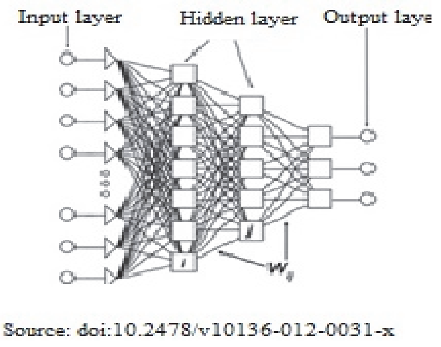


**Figure 2** Network layers

Perceptrons, which were invented by Frank Rosenblatt in 1958, are the simplest type of neural network. They consist of a single neuron and are used for binary classification tasks. Here, '$n$' represents the number of features in our dataset. The process of passing data through a neural network is known as forward propagation, and it can be explained in the following three steps when using a perceptron.

Step 1: For each input, multiply the input value $y_i$ by the weights $w_i$ and sum up all the multiplied values Eq. (1). Weights represent the strength of the connection between neurons and determine how much influence an input will have on the neuron's output. If weight $q_1$ has a higher value than weight $q_2$, then it will have a greater impact on he output than $q_2$.

$$\sum = (y_1 \times w_1) + (y_2 \times w_2) + \ldots + (y_n \times w_n) \tag{1}$$

The input and weight vectors can be represented as $y = [y_1, y_2, \ldots, y_n]$ and $w = [w_1, w_2, \ldots, w_n]$ respectively, and their vector product is given by Eq. (2):

$$y, w = (y_1 \times w_1) + \ldots + (y_n \times w_n) \tag{2}$$

So, the sum Eq. (3) is equal to the product of the vectors $x$ and $w$.

$$\sum = y \times w \tag{3}$$

Step 2: Add a bias term $b$ to the sum of multiplied values, and let's call this $z$ Eq. (4). Bias, also known as a shift, is necessary in most cases to shift the entire activation function to the left or right in order to generate the desired output values.

$$z = y \times w + b \tag{4}$$

Step 3: Pass the value $z$ through a non-linear activation function. Activation functions are used to introduce non-linearity into the output of neurons, without which the neural network would be just a linear function. Moreover, they have a significant impact on the learning speed of the neural network. Perceptrons have a step function as the activation function. The sigmoid function, also known as the logistic function, is used as our activation function Eq. (5).

$$\dot{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{5}$$

where $\sigma$ represents the sigmoid activation function, and the output obtained after forward propagation is known as the predicted value $\hat{y}$, [3].

## 3 LEARNING ALGORITHM

The learning algorithm consists of two main parts: backpropagation and optimization. Backpropagation, which stands for backward error propagation, involves calculating the gradient of the loss function with respect to the weights. However, the term "backpropagation" is commonly used to refer to the entire learning algorithm. The backpropagation process in the perceptron is explained in the following two steps: Note: The text appears to be incomplete. If there is more information to include, please provide the complete text, and I'll be happy to continue the explanation.

Step 1: To evaluate the deviation from the desired solution, a loss function is employed. In regression problems, the mean squared error is typically chosen as the loss function, while for classification problems, cross-entropy is commonly used. Let's focus on a regression problem where the loss function Eq. (6) is mean squared error, which involves squaring the difference between the actual ($y_i$) and predicted values ($\hat{y}_i$).

$$MSE_i = \left(y_1 - \dot{y}_i\right)^2 \tag{6}$$

The loss function Eq. (7) is calculated for the entire training dataset, and its average is referred to as the cost function *C*.

$$C = MSE = \frac{1}{n}\sum\nolimits_{i=1}^{n}\left(y_i - \dot{y}_i\right)^2 \tag{7}$$

Step 2: To determine the optimal weights and biases for our perceptron, we must comprehend how the cost function changes concerning the weights and biases. Gradients are utilized to represent the rate of change of one quantity concerning another. In this scenario, it was calculated the gradient of the cost function *C* concerning the weight $w_i$ by taking partial derivatives. Since the cost function is not directly dependent on the weight $\mathrm{w_i}$, the chain rule of differentiation Eq. (8) will be aplied to compute the gradient:

$$\frac{\partial c}{\partial q_i} = \frac{\partial c}{\partial \dot{y}_i} \times \frac{\partial \dot{y}_i}{\partial z} \times \frac{\partial z}{\partial q_i} \tag{8}$$

Therefore, Eq. (9) is obtained Eq. (9):

$$\frac{\partial c}{\partial q_i} = \frac{2}{n} \times \mathrm{sum}\left(y_i - \dot{y}_i\right) \times \sigma(z) \times \left(1 - \sigma(z)\right) \times x_i \tag{9}$$

### 3.1 Optimization

Optimization involves selecting the best elements from a set of available alternatives, which, in our case, refers to determining the optimal weights and biases for the perceptron. For this purpose, the gradient descent optimization algorithm, will be used, which updates the weights and biases proportionally to the negative gradient of the cost function concerning the corresponding weight or bias. To control the extent of weight and bias updates, we introduce a hyperparameter known as the learning rate (*α*). The learning rate influences the step size taken in the direction opposite to the gradient during the optimization process. The weights and bias are updated using the following equations Eq. (10) and Eq. (11), and the backpropagation and gradient descent process are repeated iteratively until convergence is achieved [4].

$$q_i = q_i - \left(a \times \frac{\partial c}{\partial q_i}\right) \tag{10}$$

$$d = d - \left(a \times \frac{\partial c}{\partial d}\right) \tag{11}$$

## 4 EXPERIMENTAL DESIGN

In the introductory part of this paper, the functioning of a single neuron was explained. However, these fundamental concepts can be adapted and applied to all types of neural networks with some modifications. The following, will delve into the experimental design, starting with a discussion on how to create a neural network in MATLAB. Subsequently, we will explain the process of recognizing a healthy or diseased kidney image using that network.

This section explains the process of designing our MATLAB code and presents the results obtained using Simulink. Additionally will be provided a brief introduction to the neural network tools available in MATLAB [8-12].

```
%% Introduction
% Brief introduction to the Neural Network Toolbox for MATLAB
% Simple example: how to prepare training data, create a neural
network,
% train and simulate it.
clc
clear all
%%
% Create the training input vector:
P = [0 1 2 3 4 5 6 7 8 9 10];
% Create the training output vector (target vector):
T = [0 1 2 3 4 3 2 1 2 3 4];
% The values in the P and T vectors form pairs: T(n) is the expected
output value for input P(n).
%%
% Create a 'clean' network
net = newff([0 10], [5 1],{'tansig','purelin'});
% View the parameter description of the newff function by typing:
help newff
% Set the number of training epochs:
net.trainParam.epochs = 200;
% and train the network:
[neto, tr, Y, E] = train(net, P, T);
% View the parameter description of the train function by typing:
help train
% Now you can simulate the network (check the response of the neural
network):
A = sim(net, P);
% The neural network can be saved to a file like any other variable or
object in MATLAB.
%%
% Explore the transfer functions available within the toolbox:
n = −55:0.5:55;
plot(n, hardlim(n), 'r + :');
```

Now, some key lines of code will be explained. In line 7, a 1 × 11 vector is created for the training input. Subsequently, in line 9, a 1 × 11 vector is created for the output. Later, in line 13, a feedforward neural network is created with an input vector of [0 10] and an output vector of [5 1]. The first activation function is set to 'Tansig', representing the hyperbolic tangent sigmoid transfer function (Fig. 3), while the second activation function is set to 'purelin', representing the linear transfer function (Fig. 3). The number of epochs is specified as 200 using the line Net.trainParam.epochs = 200. This indicates the number of training iterations, and in this case, the neural network is trained for 200 epochs.

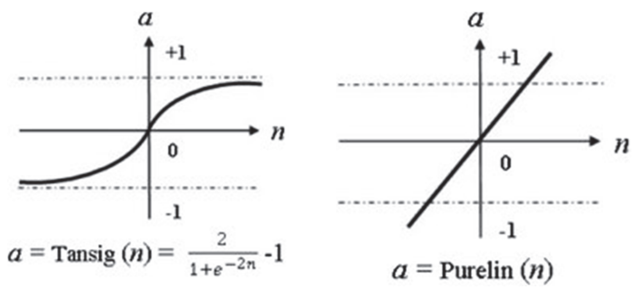
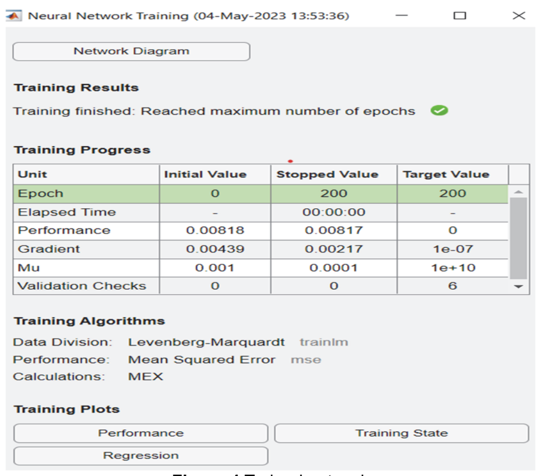


**Figure 3** Tansig and purelin functions

In the line [net, tr, *Y*, *E*] = train(net, *P*, *T*); the neural network has been trained for 200 epochs. Now, we can assess its accuracy and simulate the network using *A* = sim(net, *P*). The simulated network can be seen in Fig. 4.



Neural Network Training (04-May-2023 13:53:36)

Network Diagram

**Training Results**

Training finished: Reached maximum number of epochs ✓

**Training Progress**

| Unit | Initial Value | Stopped Value | Target Value |
|---|---|---|---|
| Epoch | 0 | 200 | 200 |
| Elapsed Time | - | 00:00:00 | - |
| Performance | 0.00818 | 0.00817 | 0 |
| Gradient | 0.00439 | 0.00217 | 1e-07 |
| Mu | 0.001 | 0.0001 | 1e+10 |
| Validation Checks | 0 | 0 | 6 |

**Training Algorithms**

Data Division: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

**Training Plots**

Performance | Training State | Regression

**Figure 4** Trained network

The best performance will be achieved in the 200 epochs (Fig. 5). Therefore, our epoch number can be set to 200 for improved efficiency.

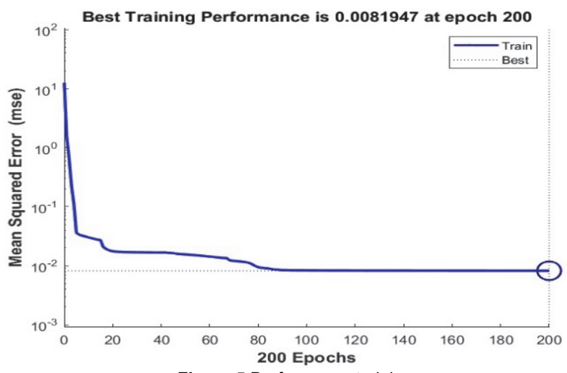


**Figure 5** Performance training

## 5 IMAGE RECOGNATION

In this project, goal is to provide a neural network with 20 different images of size 227 × 227 pixels in binary code format, ten of which are images of healthy and ten of diseased kidneys (with various kidney diseases). Then, the network is trained with the aim of accurately predicting which kidney disease can develop into cancer, all in order to treat the patient on time, at the stage when better results can be achieved with the treatment.

The images have a matrix of 227 × 227 × 3, which means they are three-dimensional. So these are color photos (the third dimension indicates color). In order to simplify the analysis, three-dimensional images are converted into two-dimensional ones, that is, into black and white. The number of pixels is represented by 227 × 227.

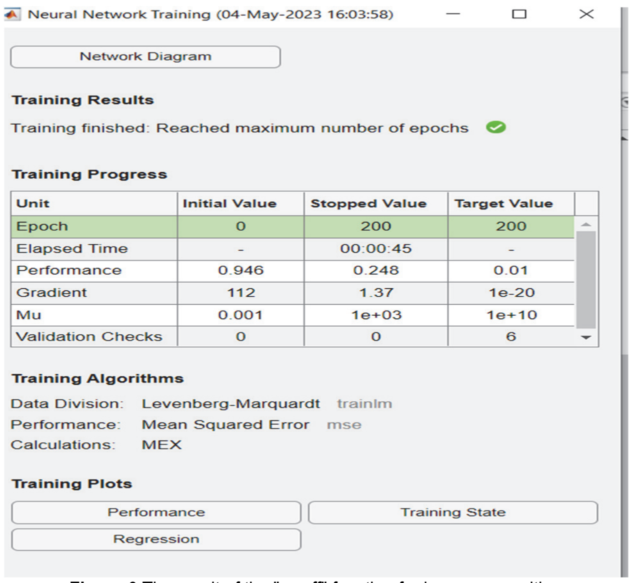
As mentioned before, each image has a dimension of a 227 × 227 matrix, and it uses a total of 20 images. Each pixel serves as an input to the ANN, resulting in a large number of pixels, which significantly burdens computer resources and leads to the fact that too much time is needed for each epoch in ANN. In order to reduce the size of each image the impres size function of Matlab will be used.

Our new size is a matrix of size 20 × 20, which means 400 inputs per image. Finally, the network is ready to start training and simulation. Now, let's discuss the results.

The results of network training are usually presented in the form of a table (Fig. 6) that includes several important metrics that give us insight into how well the network has been trained on the given dataset. Specifically, for each training epoch (in this case, the initial and final epochs 0 and 200), the following metrics are shown:
Elapsed Time: The time elapsed from the start of training to the current epoch (in this case, 0 seconds and 45 seconds).
Performance: The network's performance on the current dataset (in this case, 0,946 at the beginning of training and 0,248 at the end of training). This value is usually calculated as the mean squared error (MSE) or mean absolute error (MAE) between the network's outputs and the expected outputs on the current dataset, [13]. A lower performance value typically indicates better training of the network.
Gradient: The value of the gradient of the error function (in this case, 112 at the beginning of training and 1,37 at the end of training). The gradient is used for updating the network's weights and its value changes during training. A smaller gradient value indicates better training of the network.
Mu: The value of the Mu parameter is used in adaptive weight updating (in this case, 0,001 at the beginning of training and 1e + 03 at the end of training). The value of this parameter also changes during training.
Validation Checks: The number of validation checks performed up to the current epoch (in this case, 0 at the beginning and 6 at the end of training). Validation is used to prevent overfitting of the network to the training dataset and is typically done on an independent dataset. If the number of validation checks is too high, it can indicate that the network has memorized the mentioned dataset and fails to generalize well the new data.

In the context of performance, the statement "Best Training Performance is 0,24759 at epoch 200" means that after 200 epochs of training the neural network, the lowest error value achieved between the target values (*T*) and the predicted values (*Y*) on the training set was 0,24759. This means that the network has learned to correctly map the input values (*P*) to the target values (*T*) with an error of 0,24759; which is considered a quite good result. This is specific to this code and input data.

Generally speaking, the best performance of a neural network can vary depending on various factors such as the specific problem being solved, the architecture of the neural network, the chosen hyper parameters, and other relevant considerations (Fig. 7).


Neural Network Training (04-May-2023 16:03:58)

Network Diagram

**Training Results**

Training finished: Reached maximum number of epochs ✓

**Training Progress**

| Unit | Initial Value | Stopped Value | Target Value |
|---|---|---|---|
| Epoch | 0 | 200 | 200 |
| Elapsed Time | - | 00:00:45 | - |
| Performance | 0.946 | 0.248 | 0.01 |
| Gradient | 112 | 1.37 | 1e-20 |
| Mu | 0.001 | 1e+03 | 1e+10 |
| Validation Checks | 0 | 0 | 6 |

**Training Algorithms**

Data Division: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

**Training Plots**

**Figure 6** The result of the "newff" function for image recognition

For optimizing the weights of neural networks, one of the commonly used algorithms is the gradient descent algorithm [14]. This algorithm leverages the gradient of the loss function to iteratively adjust the network's weights in a way that aims to minimize the overall loss or error of the network's predictions.

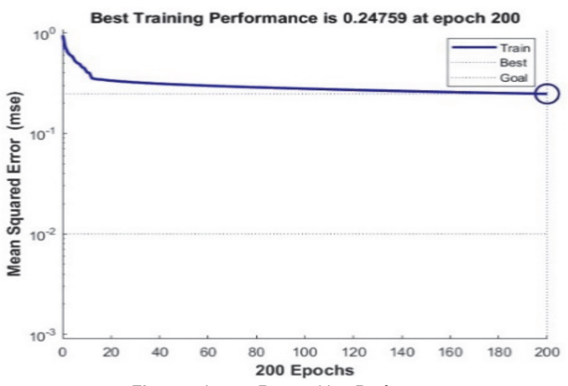


**Figure 7** Image Recognition Performance

In this case (Fig. 8), the value of Gradient = 1,3674 represents the gradient of the loss function in the last epoch of training. It indicates how much the loss has changed concerning the network weights during the final epoch of training. A smaller gradient value suggests that the network is closer to achieving optimal performance. Mu = 1000 denotes the learning rate in the last epoch of training. The learning rate is a parameter that governs the extent to which the network weights are updated in each optimization step. A higher learning rate can result in more substantial weight updates, potentially leading to faster convergence but risking overshooting the optimal solution. Validation Checks = 0 indicates that the number of times the network's performance on the validation set is checked during training is zero. In other words, the validation set's performance is not assessed during training to monitor for potential overfitting. This means that the training process does not include any stopping criteria based on the validation set's performance. Overall, these metrics provide valuable insights into the training process, helping to understand how the neural network is adapting and whether it is making progress towards the desired solution. Monitoring and interpreting these metrics can guide the optimization of the neural network and improve its performance on the specific task at hand.

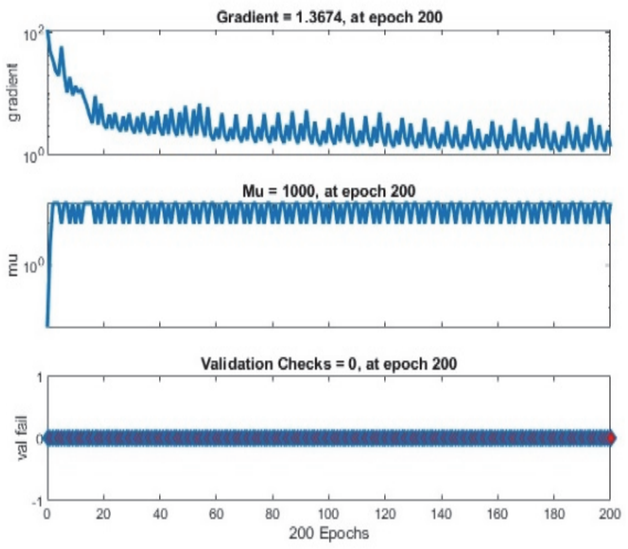


**Figure 8** Training epochs

The value of *R* obtained after enabling the regression option is the coefficient of determination, (Fig. 9) commonly denoted as *R*-squared ($R^2$).

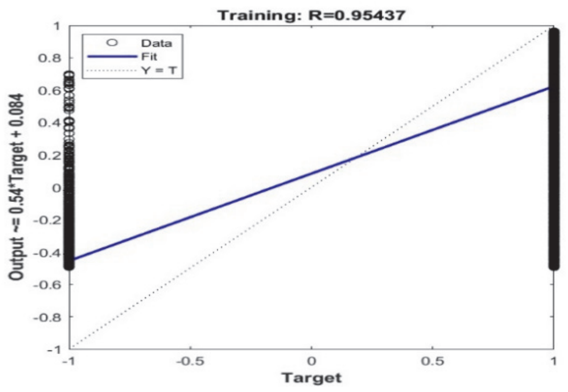


**Figure 9** Regression results

The coefficient $R^2$ is utilized to evaluate the quality of fit of a regression model to the training data. $R^2$ assumes values between 0 and 1, where 0 signifies that the model fails to explain the variability of the target variable, and 1 denotes that the model perfectly explains the variability of the target variable. In this case, the value $R^2$ = 0,95437 indicates that our model adTuately explains the variability of the target variable. Our trained network attained the highest accuracy in detecting kidney disease (hydronephrosis) during epochs 242 to 250 out of the 1000 epochs tested. This period of training resulted in the best performance for detecting the specific kidney condition. Images are divided into two classes and classified accordingly. Firstly, the paths to the folders containing the data for each class (healthy and diseased) are established. Subsequently, functions from the Image Processing Toolbox are employed to prepare the data for training. The data is then consolidated into a single dataset and split into separate sets for training and testing purposes.

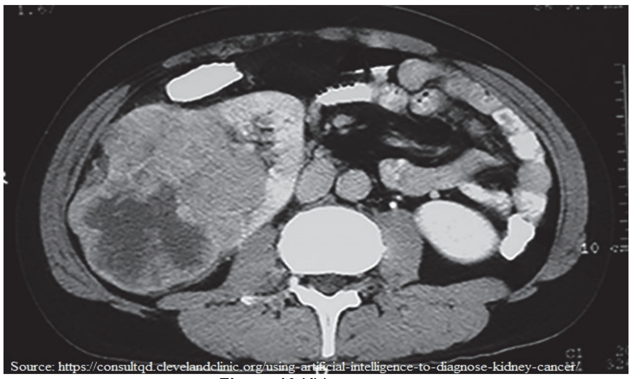

Source: https://consultqd.clevelandclinic.org/using-artificial-intelligence-to-diagnose-kidney-cancer/

**Figure 10** Kidney cancer

Following that, the architecture of the neural network is defined, incorporating layers for convolution, normalization, and pooling filtering, as well as fully connected and softmax classification layers. The network's training options are specified, including the maximum number of epochs, batch size, and optimization algorithm. The network undergoes training using the training set, and after each pass through the training set, it is evaluated on the testing set.

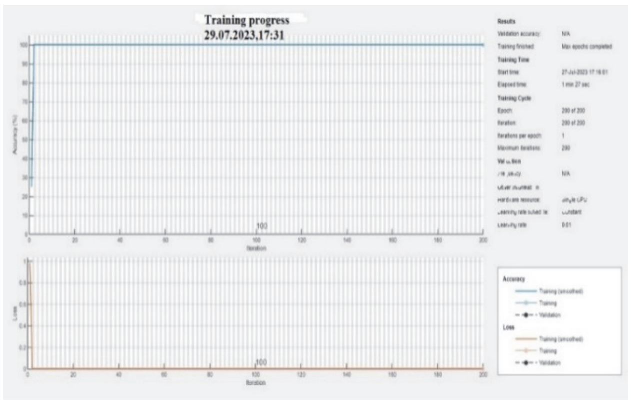


**Figure 11** Training Results for 200 epochs

Finally, graphs presenting the training loss and accuracy throughout the learning process are displayed. These values are interpreted using an optimization algorithm, such as stochastic gradient descent (SGD), with the following explanations:
Epoch: This indicates the number of times the optimization algorithm (SGD) has iterated over the entire training dataset. Each pass through the dataset is referred to as one epoch, and in this case, it is 200.

Other parameters like Elapsed Time, Performance, Gradient, Mu, and Validation Checks have been previously explained.

After conducting testing evaluation for 1000 epochs, the most promising research results for hydronoze kidney disease were achieved at epoch 248, as illustrated in Fig. 12.

In Fig. 12, it is indicated that the best validation performance (lowest loss) was achieved at epoch 242, with a value of 0,00099728. Validation performance is typically used to assess the effectiveness of the model during training and measures how well the model generalizes to unseen data. Therefore, a lower validation performance indicates a better model, as it suggests better generalization to unseen data. In this case, the model achieved very good results with a very low loss on the validation set in the range of epochs 242 to 248, indicating a strong ability to generalize. The gradient (Fig. 13) refers to the value of the gradient of the network's loss function at a specific epoch, which can help assess the speed and stability of network training. In this case, the Gradient value at epoch 248 is 9,3358e−05.

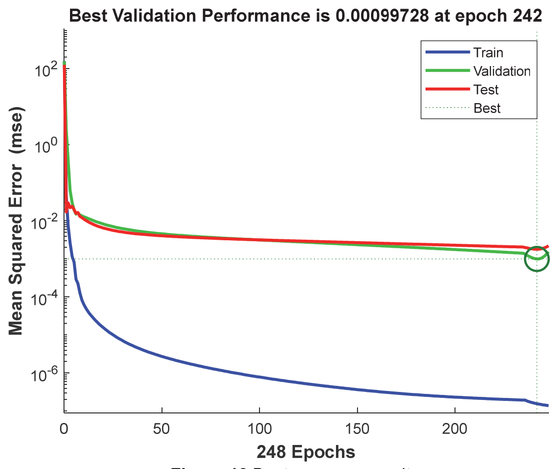


**Figure 12** Best accuracy results

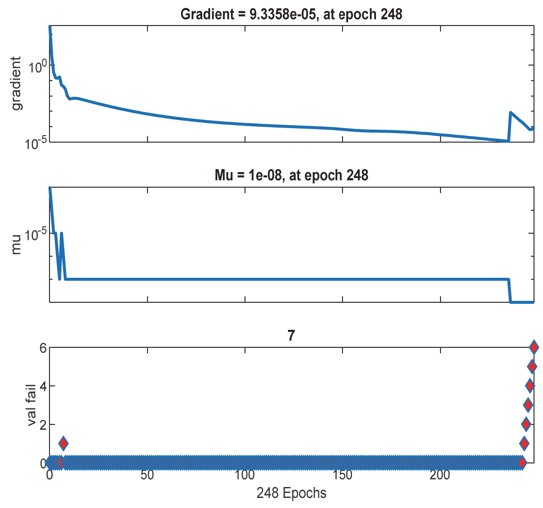


**Figura 13** Gradient visualization

Mu refers to the value of the learning rate parameter used to adjust the network weights during training. The value of this parameter affects the convergence speed and training stability. In this case, the Mu value at epoch 248 is 1e−08. Validation Checks represents the number of validation checks performed during network training, which indicates how many times the network's performance on the validation set was evaluated to prevent overfitting.

These metrics provide important insights into the training process and help in understanding the performance and behavior of the neural network throughout the training epochs.

Validation checks are typically conducted after each epoch to assess the network's accuracy on validation data and can be used to monitor network performance during training. In this case, 6 validation checks were performed at epoch 248. The plots display the network's performance during training in terms of the loss function and accuracy on the training and validation sets. Based on these plots, one can assess whether the network is overfitting or underfitting and estimate its accuracy on unseen data.

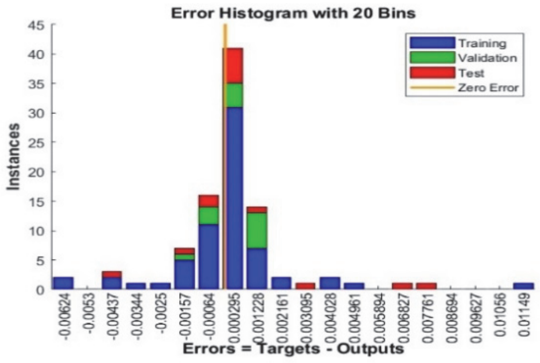


**Figure 14** Error Detection visualization during neural network testing

Image 14 displays the errors detected during training. The error histogram with 20 bins represents a chart that illustrates the distribution of network errors after evaluation on the testing set. This chart uses a histogram with 20 bins (columns) to display the frequency of error occurrences within specific ranges. The horizontal axis represents the error ranges (bins), while the vertical axis represents the number of instances with that error. The higher the bin, the greater the number of instances that have an error within that range. This visualization allows for easy understanding of how errors are distributed in testing and their magnitude. If the histogram shows a uniform distribution, it suggests that the network is reasonably well-trained, as errors are evenly spread across all bins. However, if the histogram exhibits a high degree of asymmetry, it indicates that there may be an issue with training, which could be due to data imbalance or other problems in the model. Fig. 15 illustrates the training and validation detections for the obtained training results.

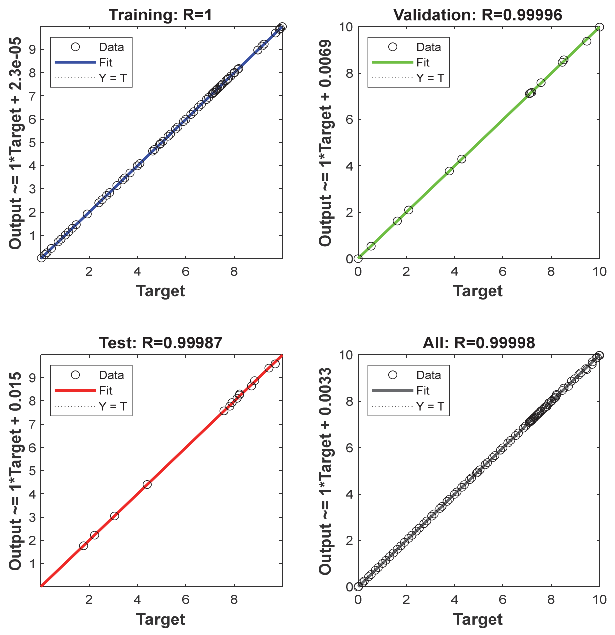


**Figure 15** Training and validation

These graphs use the *R*-value to assess how well the neural network generalizes to new data. Training: The *R*-value is equal to 1, indicating that the neural network has perfectly fit the training data. Validation: An *R*-value of 0,99996 indicates that the network has adapted very well to the validation set and generalizes well to new data. Test: An *R*-value of 0,99987 indicates very good generalization of the network to new data. All: An *R*-value of 0,99998 indicates excellent generalization of the neural network to new data and suggests that the network has adapted well to the dataset. Training loss represents the difference between the expected and actual outputs of the neural network, and the goal of training is to minimize this loss.

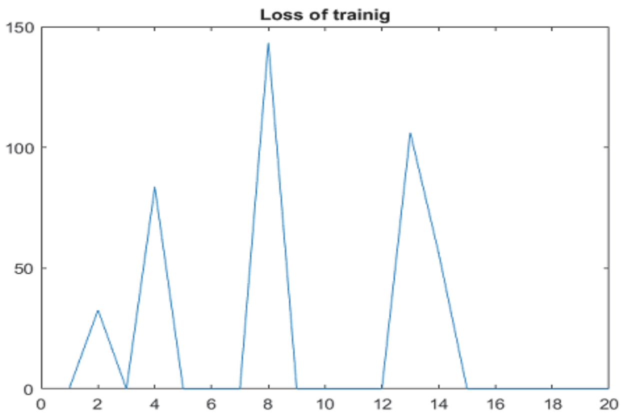


**Figure 16** Training loss

Therefore, the graph shows how well the neural network is learning during the training process and how the loss decreases over time. The lower the loss, the better the accuracy of the network (Fig. 16). Accuracy represents the percentage of correctly classified samples in the training set (Fig. 17).

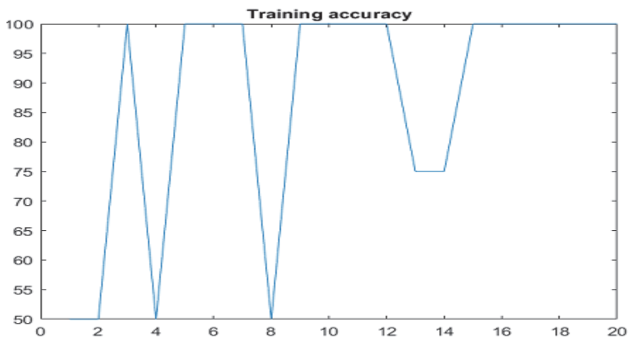


**Figure 17** Training accuracy

The graph illustrates how accuracy changes over time, i.e., during training iterations. As the neural network learns, the accuracy should increase. Therefore, the higher the accuracy, the better the training of the network. Overall, these two graphs provide insight into how well the neural network is learning during the training process and how its performance changes over time.

## 6 CONCLUSION

In this study, the application of artificial neural networks and machine learning algorithms is explored for the recognition of kidney conditions and the detection of kidney cancer. The goal was to utilize neural networks that simulate the functioning of the human brain to achieve an effective approximation of recognizing images of healthy individuals and their kidney conditions, as well as kidney conditions affected by various diseases. By using samples in the form of kidney condition images and applying machine learning algorithms, significant results are achieved in the detection of kidney cancer. The article demonstrates that an improvement in the possibility of early detection of kidney cancer is achieved by increasing the number of epochs in the training sequence. One shortcoming of this research is the relatively small number of analyzed images, ten for healthy and ten for diseased kidneys. The results of this research have significant implications in medicine, particularly in clinical practice as they serve as an aid in the early diagnosis of kidney diseases. This study demonstrates that the application of artificial neural networks and machine learning algorithms can be highly beneficial in enhancing the quality of medical diagnostics. Therefore, further research in this area is recommended to advance the methods of recognizing and treating kidney diseases.

## 7 REFERENCES

[1] See: https://askabiologist.asu.edu/neuron-anatomy

[2] Singh, V., Asari, V. K., & Rajasekaran, R. (2022). A Deep Neural Network for Early Detection and Prediction of Chronic Kidney Disease. *Diagnostics*, *12*(1), 116. https://doi.org/10.3390/diagnostics12010116

[3] Amato, A., Lopez, E. M., Pena Mendez, P., Vanhara, A. H., & Havel, J. (2013). Artificial neural networks in medicine diagnosis. *J Appl Biomed*, *11*(2), 47-58. https://doi:10.2478/v10136-012-0031-x

[4] Chen, W., Zeng, R., Jin, Y., Sun, X., Zhou, Z., & Zhu, C. (2022). Artificial Neural Network Assisted Cancer Risk Prediction of Oral Precancerous Lesions. *Bio Med research international*, 7352489. https://doi.org/10.1155/2022/7352489

[5] Qin, J., Chen, L., Liu, Y., Liu, C., Feng, C., & Chen, B. (2020). A Machine Learning Methodology for Diagnosing Chronic Kidney Disease. *IEEE Access*, *8*, 20991-21002. https://doi: 10.1109/ACCESS.2019.2963053

[6] Chen, G., Chenguang, D., Li, Y., & Hu, X. (2020). "Prediction of Chronic Kidney Disease Using Adaptive Hybridized Deep Convolutional Neural Network on the Internet of Medical Things Platform. *IEEE Access*, *8*, 100497-100508. https://doi: 10.1109/ACCESS.2020.2995310

[7] Chakrapani, C. & Raj, S. (2019). Detection of Chronic Kidney Disease Using Artificial Neural Network. *International Journal of Applied Engineering Research, 14*(10) 145-149

[8] Brownlee, J. (2019). *How to Choose Loss Functions When Training Deep Learning Neural Networks*.

[9] Doshi, S. (2019). *Various Optimization Algorithms to Training Neural Network*.

**Contact information:**

**Dario GALIĆ**, assistant professor
(Corresponding author)
Faculty of Dental Medicine and Health Osijek,
31000 Osijek

**Zvezdan STOJANOVIĆ**, full professor
European University Brcko,
76100 Brčko
E-mail: zvezdan.stojanovic070@gmail.com

**Elvir ČAJIĆ**, phd candidate
European University Brcko,
76100 Brčko
E-mail: ecajic86@gmail.com