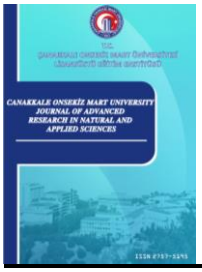
e-ISSN 2757-5195



Çanakkale Onsekiz Mart University

Journal of Advanced Research in Natural and Applied Sciences

Open Access

doi.org/ 10.28979/jarnas.890552 2021, Vol. 7, Issue 1, Pages: 12-34 dergipark.org.tr/tr/pub/jarnas

The Use of Graph Databases for Artificial Neural Networks

Doğa Barış Özdemir[1], Ahmet Cumhur Kınacı[2]

[1]Department of Computer Engineering, Faculty of Engineering, Çanakkale Onsekiz Mart University, Çanakkale, Turkey

[2]Department of Computer Engineering, Faculty of Engineering, Çanakkale Onsekiz Mart University, Çanakkale, Turkey

Article History

Received: 08.05.2020

Accepted: 14.12.2020

Published: 20.03.2021

Research Article

Abstract − Storing and using trained artificial neural network (ANN) models face technical difficulties. These models are usually stored as files and cannot be run directly. An artificial neural network can be structurally expressed as a graph. Therefore, it would be much more useful to store ANN models in a database and use the graph database as this database system. In this study, training and testing stages of ANN models are provided with software that will allow multiple researchers to conduct joint research on ANN models. The developed software platform is aimed to increase the representation power of the currently used methods by transferring the models developed in the popular ANN frameworks used today. With the study conducted, even someone who has started learning artificial neural network models from scratch will see the process and can visually develop their own model. When models are stored in the graph database, it will be easier to making versions and observing how the model grows. In addition, data to be input and output to the model can be stored in this database, also. In order to feed ANN models with input data and produce outputs, the graph database's own query language was used. This eliminates the dependency on another software library.

Keywords *– Collaborative neural network model training, database based artificial neural networks, graph-based artificial neural networks, representation of artificial neural networks, visualization of artificial neural networks*

1. Introduction

Most of the advances in artificial intelligence are based on statistical models. The majority that provides the most comprehensive advances in statistical models is artificial neural networks. The artificial neural network is the model in which the learning mechanics of the human brain is applied roughly in the computer environment. The artificial neural network has computational units called neurons as in the human brain. Neurons are connected by structures called synapses with weights. The calculation in neurons is provided by multiplying the information in the neuron with the weight in the synapse. In artificial neural networks, with the bond of synapses between neurons and the formation of layers of neurons, it creates network models as in the human brain. With the study of McCulloch & Pitts (1943) first artificial neuron proposed mathematically. Artificial neurons also called as Threshold Logic Units. With the development of artificial neurons, a base mathematical method for neural network models has been prepared, and the artificial neural network era has been started. With the study of Rosenblatt (1958) first artificial neural networks were founded. The first artificial neural network called as Perceptron. Perceptron algorithm learning is linear. In not linear separable data space, Perceptron can't learn the classes of data space. Perceptron learning is based on Hebb (1949) study. By saying that information is stored in the links between neurons, Hebb (1949) suggested that the actions that stimulate or inhibit the neuron are in the information between the inputs and the bonds. The basic

[1] dogabarisozdemir@gmail.com

[2] cumhur.kinaci@comu.edu.tr

principle of Hebb theory is to strengthen the bond between two neurons if they are active at the same time and weaken the bond if they are not active at the same time.

Minsky & Papert (1969) showed that in response to the misconception that Perceptrons, formed by the widespread use of Perceptrons, can solve all kinds of problems, artificial neural networks can only solve linear problems. In their work, they aimed to learn the XOR logic gate as an example. With the failure of the model in the example, they caused the era of artificial neural networks to pass into stagnation. After 10 years of recession, when Kohonen (1982) offered a method for self-organizing neural networks, artificial neural networks began to gain popularity again. At the same time, Hopfield (1982) and Hopfield (1984) also contributed to popularization by laying the foundations of new models with his contributions to the mathematical foundations of artificial neural networks in his studies. Thus, Hopfield has shown that artificial neural networks can be generalized and solutions can be produced to problems that are difficult to solve with computer software. Hopfield solved the difficult to solve the Travelling Salesman Problem with the artificial neural network he established. Rumelhart, Hinton & Williams (1985) developed the capabilities of artificial neural networks by introducing the concept of error backpropagation in artificial neural networks. Modern neural networks are divided into input, hidden and output layers based on multilayer perceptrons with backpropagation (Rumelhart et al., 1986). These models are also called recurrent neural networks.

While modern artificial neural networks perform learning, they multiply the weights in the connections between neurons by the input and achieve results by adding the bias value on it. The bias value can be a fixed value or a dynamically changing value. To choose a fixed value, the researcher must know the data space well. Even if the data space is well known, the designed ANN may not achieve the desired result. It is also called the learning rate instead of bias. The part of summing these results and writing them to the neuron is called the summation function. Although there are different types of sum functions, the most common is the summation of all calculated inputs in links. With this calculated result, activation functions decide whether the neuron will be active or not and what kind of output it will produce for the input. The most important feature of the activation function is to ensure the nonlinearity of the artificial neural network. While activation functions take on this important task, besides the selection of nonlinear functions, their derivatives should be calculated with low computational costs. Binary Step, Linear, Sigmoid (Cybenko, 1989), Tanh, ReLU (Nair & Hinton, 2010), Leaky ReLU (Maas, Hannun & Ng, 2013), SoftMax (Goodfellow, Courville, Bengio, 2016) activation functions are the most commonly used activating functions. The activation functions vary according to the data types and output types in the data space, as well as the architecture of the ANN model, established. While using the Linear function while solving linear data space problems, the Binary Step function is used for the neuron output that needs to be binary. It is often used because the sigmoid function is nonlinear and its continuous derivative is available. For example, Sigmoid function is frequently used in multilayer perceptron models. The sigmoid function produces outputs between zero and one. When calculations are made with extreme input values in sigmoid functions, obtaining outputs converging to zero causes gradient loss problem. This approach provides slow learning and prevents getting closer to the desired global minimum point by ensuring that the local minimum points are stuck. Unlike the Sigmoid function, Tanh function obtains output between -1 and +1 values. Although a larger output area is obtained compared to the Sigmoid function, still the disappearing gradient problem cannot be avoided with Tanh function. The ReLU function takes inputs between 0 and infinite values, and its output is either 0 or 1. The calculation cost of ReLU is less than the Sigmoid and Tanh functions. While ReLU accelerates the calculations with inputs in the zero value region, it may cause the learning not to occur because the derivative values in the region are zero. This problem is called the dying neuron or dying ReLU problem. In response to this problem of ReLU, Leaky ReLU function is revealed to ensure that the outputs are not zero. In Leaky ReLU, values between negative infinity and positive infinity are obtained. In Leaky ReLU, it is ensured that a zero result is not obtained by adding the leakage value. Thus, it is ensured that

neurons do not die. Softmax preferred in multi-class ANNs that have more than two class such as Sigmoid function. Softmax gives the probability of its result between 0 and 1 with other classes.

There are three main types of learning in artificial neural networks, depending on the problem to be solved and the data set. Each type of learning has advantages as well as disadvantages. With supervised learning, the researcher must have a command of the problem and know the intended outcomes after the solution of the problem (Öztanır, 2018). The desired outputs and the inputs of these outputs must be marked by the researcher. While supervised learning is being executed, the marked data sets are divided into two parts as training and testing (Tosun, 2007). Generally, the training data set covers a bigger part than testing data set. It aims to reveal a pattern by making error calculations with its proximity to the targeted outcomes by learning with the artificial neural network training data set. The test data set is used to understand the performance of models that are thought to have a high enough performance in data they have never seen. It is a frequently used and generally accepted learning model in classification and estimation problems when the problem space is known. The unsupervised learning method is used in the solution of clustering problems where the problem space is not known and cannot be marked. It aims for the neural network to self-organize to identify clusters and split data (Çuhadar & Kayacan, 2005). Researchers become able to classify results obtained in unsupervised learning by comparing them with real-world problems. The main purpose is to separate the data rather than the solution to the problem. In reinforcement learning, a model in the training process presents the outputs to the researcher and waits for the markup. It is used in situations where marking could not be done before. Model learning takes place in a long time as it needs expert support for each output. It is used in fields such as medicine where critical decisions are made. For example, when it is desired to determine the disease with the test results of a patient, the model examines the data and decides then shows the doctor. The doctor examines the detection and provides feedback to mark it as true or false. This model, which has been learned with support over a long time, can obtain more accurate outputs.

Until now, not all problems have been solved with a single artificial neural network type. For this reason, many different models with proven problem-specific advantages have been developed and continue to be developed. Widrow & Hoff (1960) revealed the learning method of feed-forward adaptive neural neuron with their study. Adaline as known as the adaptive linear neuron is the simplest of all neural networks and uses reinforcement for learning. It is insufficient to solve complex problems due to its linear operation. The multi-layered form of Adaline is called Madaline. Madaline consists of three layers as input, hidden and output layer. Madaline consists of a combination of many Adaline neurons. Madaline has been used for a wide variety of applications such as voice recognition, character recognition, weather prediction, and adaptive control (Widrow & Lehr, 1990). Due to the inability to backpropagation, its use has not become widespread after its first release. Tank & Hopfield (1986) applied the logic of showing the state of the brain in a box in artificial neural networks with their study. The Hopfield model, with its single layer and fully connected neurons, enables the system to converge to a state with its current inputs. With this method, which is self-organized and mimics the state of the brain during learning, solutions are sought for optimization problems. Carpenter & Grossberg (1990) conduct unsupervised learning for classification problems with the adaptive resonance theory they developed. With the adaptive resonance theory, it is aimed not to lose the previously learned information on neural networks. It contains two layers as input and output, but the number of neural networks in the layers changes dynamically. Ackley, Hinton & Sejnowski (1985) introduced the concept of the energy function network with the Boltzmann machine they developed. The Boltzmann machine, which differs from the Hopfield network that it contains hidden layers, works by aiming at minimizing the energy function. It contains a layer to which all neurons are fully connected. The restricted Boltzmann machine model introduced by Smolensky (1986) as Harmonium. Unlike the Boltzmann machine, the restricted Boltzmann machine does not contain links within each layer and contains hidden layers with feedback input. With these restrictions, more effective learning has been achieved. Moody & Darken (1989) presented a generalizable convergence model with their study. The model is called a radial basis function.

The difference from the feed-forward network is that it has a fast learning speed and can only give true or false values. It is used in time series problems and system control areas. Unlike other neural networks, Kohonen (1990) does not keep neurons in the system by keeping the connection weights in vectors. Thus, new neurons can be included in the system and dynamism is given to the network. The proximity of the weight vectors in neuron bonds to the winning vector is calculated by the dot product. This method is called the learning vector quantization. Hochreiter & Schmidhuber (1997) carried recurrent neural networks forward by working on a long short-term memory model. LSTM tries to obtain meaningful outputs with data in old and new times by including input, output and memory layers. They have proven successful in speech recognition and writing recognition. In their study, Cho et al. (2014) add a forgetting layer to LSTM models and determine how much past data will be remembered and when it will be forgotten. Thus, they developed LSTMs and brought an important model to the literature that will work with high performance in subjects such as music modelling and natural language processing. Goodfellow et al. (2014) are trying to generate new data with patterns learned through their generative adversarial networks study. It gives outputs to use the learning from the problem data to generate synthetic new data that cannot be understood by human beings. It works with performance on subjects such as ageing human face images, generating new photographs, and generating paintings. The widespread use of multi-layer networks and the increase in computing power in computers led to the emergence of artificial neural networks, which consisted of several layers, to be networks of hundreds of layers. These artificial neural networks, which can contain dozens of layers and artificial neural networks that serve different purposes of learning, are called deep artificial neural networks. Nowadays, it is seen that deep neural networks work intensely with visual data. Learning in deep networks is provided by methods under the title of deep learning. LeCun, Bottou, Bengio & Haffner (1998) made the most developed study of deep learning with convolutional neural networks. Convolutional neural network name comes from the convolution process on the data by the network.

Artificial neural networks are simply divided into three main different layers. These are the input layer, the hidden layer, and the output layer. The structures where data and relationships are kept and processed as in the human brain and artificial neural networks coincide exactly with the functioning of graphs. With Euler (1741) study the graph theory came out with the famous Seven Bridges of Königsberg problem. It is created by the nodes that define the model objects and the relationships between them. The graph is a mathematical model created to describe the relationship between objects with each other.

The problem data to be learned while performing the learning must be entered from the neurons in the input layer. Calculations should be made in hidden layers and results should be transferred to the output layer. If the results are not performing well in the output layer, the same steps should be repeated according to the type of ANN model. As the model to be learned becomes more complex, one hidden layer is not sufficient and models that need more than one layer are formed. When learning is done with traditional methods, it cannot be observed by the researcher, who makes model training due to the black box structure of the neuron and synapse data in the hidden layer (Touretzky & Pomerleau, 1989). Model design is carried out based on outputs and error rates and parameters. This approach causes researchers to work on artificial neural networks to create the model based on the output parameters and not to follow them visually.

2. Materials and Methods

Today, in various applications, data can be displayed naturally in graph structures. Graph structures can be sequential, tree, one-way, bidirectional. In artificial neural networks, nerve cells can be expressed with nodes in graph databases, and links between artificial nerve cells can be expressed in relationships in graph databases. Various studies have been conducted in the literature that deals with the subject of artificial neural networks and graph.

Nekhaev & Demin (2017) visualized the neural network and its hidden layers to solve the problem of the incomprehensibility of the operation of deep artificial neural networks. Liu (2017) significantly reduced the

search time by performing the searches on the neural network they dealt with in the article they published on the graph structure. Wang (2018) have realized better learning compared to other models in the field of reinforcement learning with the learning structure policy in graphical neural networks developed for multilayer neural networks. Muhammad & Halim (2016) examined the graphs in artificial neural networks and data representation in machine learning methods in the best visualization methods and showed them as one of the successful methods. Olden & Jackson (2002) investigated the black-box structure in artificial neural networks with their study and drew attention to the problem of finding out which nerve cell and its relationship is more valuable due to the black-box status of models in studies conducted in the environmental science of artificial neural networks. To solve the problem, they created a randomness procedure to test the statistical significance of neurons in terms of individual link weights and the overall effect of each input variable. They provided removal of link weights that did not significantly contribute to the performance of the artificial neural network and facilitated the interpretation of the direct-acting model of the input variables on the response. With the randomization procedure, they provided weightless neuron bonds and removal of insignificant input variables, thus attempting to make sense of the black box structure by reducing the complexity of the neural network. Witt, Bux, Gusew & Leser (2019) states that the black box structure is formed in artificial neural networks due to the development of the model with performance metric patterns instead of tracking the internal dynamics and working densities. Battaglia, Hamrick, Bapst, Alvaro & Razvan (2018) investigated the question of which structural representations can artificial intelligence be provided with human skills. In their study, they concluded that artificial intelligence and human skills will be achieved through gains in the rich assets, relationships and combined generalization areas in graph networks. Uwents, Monfardini, Blockeel, Gori & Scarselli (2010) made a comparison between relational neural networks and graphical neural networks, which are graphical neural network approaches, and concluded that graphical neural networks perform higher in relational data in the training of biological datasets. Studies show that the representation of artificial neural network nodes and relationships is crucial to understand how models work efficiently as a solution to the problems. In this study, presented a new method which enriches representation and working collaboratively on ANN models.

Armenta & Jodoin (2020) has mathematically examined the subject of representation of artificial neural networks in their study. They represented artificial neural networks with quivers with their work and brought a new form of representation to the literature. The mentioned studies indicate that known methods lack the representation of artificial neural networks. In their work, Choromanska, Henaff, Mathieu, Arous & LeCun (2015) cited an example of lack of representation by referring to his study. Buhrmester, Münch & Arens (2019) examined the black-box structure of artificial neural networks in deep learning and compared the studies that break black-box structures in the field of computer vision.

There are studies in the literature aiming to run artificial neural networks in databases. However, there is no artificial neural network study implemented in graph databases. It is suggested in the studies that the artificial neural networks will be expressed better by realizing them in databases. In his study, Schikuta (2008) presented an exemplary approach to implement the object-oriented method and artificial neural networks in relational databases. Using the Iris database system as a database, suggesting that all kinds of artificial neural networks can be expressed and flexibility can be added to networks with the object-oriented approach. Iris database is Knowledge-based. He separated the learning of neural networks from the architecture of the model and kept them in the database as knowledge rule sets. In the method we propose, an environment where theoretical artificial neural networks can be implemented exactly without the need for these transformations and separations is presented. Cvitkovic (2020) showed that by implementing artificial neural networks in relational databases, data extraction and feature engineering efforts in artificial neural networks will be overcome. It performs calculations on graphs by converting neural networks in relational databases into graphs. In our study, calculations can be performed directly on graphs without conversion from relational databases. It came out for the same purposes as our work and a mathematical basis of artificial

neural networks in databases is explained. Lam, Minh, Sinn, Buesser & Wistuba (2018) achieved successful results by presenting a method that performs feature extractions in artificial neural networks that they keep in relational databases. The similarities with our work are the unique key requirement for each entity in the neural network and the requirement to declare a model schema. It is difficult to determine a schema in relational databases. It focuses on static schemes. In the study we present, a model diagram can be created visually and all kinds of data can be kept. It is possible for each entity to hold different schemas dynamically in graph databases.

In their study, Yahia & Elsawi (2003) realized by keeping the neurons and connections in the ANN architecture in the relational database tables. All functions such as activation functions learning rules are also kept in the database. They realized the learning processes of ANN models by writing 4GL procedures. They wrote a static sample frontend program and created a prototype that learned OR gate in relational databases. In our work, dynamic graphs can be created with the dynamic interface. In relational databases, the relationships with entities are kept in tables and they are not the exact representation of artificial neural networks. The need for an intermediate table in many-to-many relationships will make it difficult to keep more complex models such as deep learning models in the database. In the table operations to be made in relational databases, for example, when a change is desired to be made in a table that holds the many-to-many relationship between two entities, corruption may occur in the already held data. Relational databases are not suitable for this type of dynamic and generalizable ANN representation, as their columns tend to be static and not update by their nature. There are very few studies in the literature to implement artificial neural networks in databases. Therefore, the presented study creates new areas of thinking about artificial neural networks.

Scarselli, Gori, Tsoi, Hagenbuchner & Monfardini (2008) have mathematically expressed computational processes such as learning and activation functions in the graphical representation of neural networks with their graph neural networks study. They supported the graph neural network model they presented with examples on subgraph matching, the mutagenesis, and the web page ranking. In our study, generalizable methods can be obtained by using the mathematical approaches they suggest. The point they focus on in their study is about the learning status of the data in the graph architecture. For this reason, it does not exactly coincide with our work. In our study, neural network models are defined and data input is made to neurons. In our study, there is no mention of learning on the data already in the form of graphs.

In this study, a new approach is aimed to propose to overcome current problems of traditional systems. It is the first study to implement artificial neural networks in graph databases. The study aims to increase the representation of the artificial neural networks and to enable them to develop new types of artificial neural network models by increasing their representation rather than making the processes in the most efficient neural network models. A comparison with a widely accepted library is made in Section 2.6. The developed framework is the solution to the problem that a group of researchers cannot work on the same model. The case study made offers a solution to the lack of representation in traditional methods used in artificial neural networks. With the effect of expressing artificial neural networks entirely with graphs, researchers can visually develop their models on the website. It has been shown that the black box problem can be overcome by graph representation. With the methods used in the study, the need to constantly convert the models into different formats in commonly used software frameworks has been eliminated. The developed method has been created based on these problems. For this purpose, an internet service server was established and software was designed for researchers to log in and continue joint research. To monitor and keep every moment of the artificial neural networks in the model training process, the structure where the models will be stored has been designed with the graph database. Structures were created to visualize the model and follow it live. A structure in which an artificial neural network structure can be created and transferred to the graph database has been designed. The process of transferring h5 models to the developed system, which are the model

output of artificial neural networks realized in the popular Keras framework, was designed. It has been shown that the output model of software that is accepted as Keras models can be transferred to the system can be represented in graph-based artificial neural networks. The h5 file is a data file saved in a hierarchical data format also known as HDF. It contains multi-dimensional scientific data series. H5 files are widely used in aviation, physics, engineering, finance, academic research, genomics, astronomy, electronic tools, and medicine. In the H5 file, the data are divided into two as homogeneous multi-dimensional arrays of data sets and container structures that hold data sets and other groups, making data retention simple.

2.1. Methodologies and technologies applied for the solution

Modelling of a neuron and synapse structure of artificial neural networks is exactly compatible with graph structure. In the study, neurons were used to express the node in a graph structure, and synapses to express relationships. In the study, a graph-based database system and web tool have been developed for visualizing artificial neural networks, creating them visually and solving the problem of collaboration with other researchers.

The web server has been developed so that users can access the graph database and user interface. The web interface has been developed to implement user inputs and outputs. Neo4j (2007) has been preferred as the graph database because it has the widest usage in its field, it is open to development. Since the framework created in the study is designed as an online service, a system is required to log in and register so that users sharing the same group with the user login can view the artificial neural network they have created. To log in and address the models that users work on, an interface has been created for users to log in.

As a methodology, the webserver that forms the basis of the framework was created in the study. The Angular framework was connected to the server to enable dynamic frontend operations. Researchers can create and edit ANN models with drag and drop nodes and relationships. Intuitive actions that come into our lives with touch screens are designed to make researchers think and work on model creation step easily. A database was created and connected to the server on the Neo4j server where graph database services will be used. Thus, an environment was created to make graph-based neural networks. In Figure 1, the relationships between the layers in the system architecture are shown. Neo4j database stores ANN models and serves ANNs for display. Web server executes ANN model actions which triggered from the user interface. Also, the webserver does authorization to match researchers and their ANN models. User interface handles researcher actions over ANN models and shows ANNs with graphs dynamically.

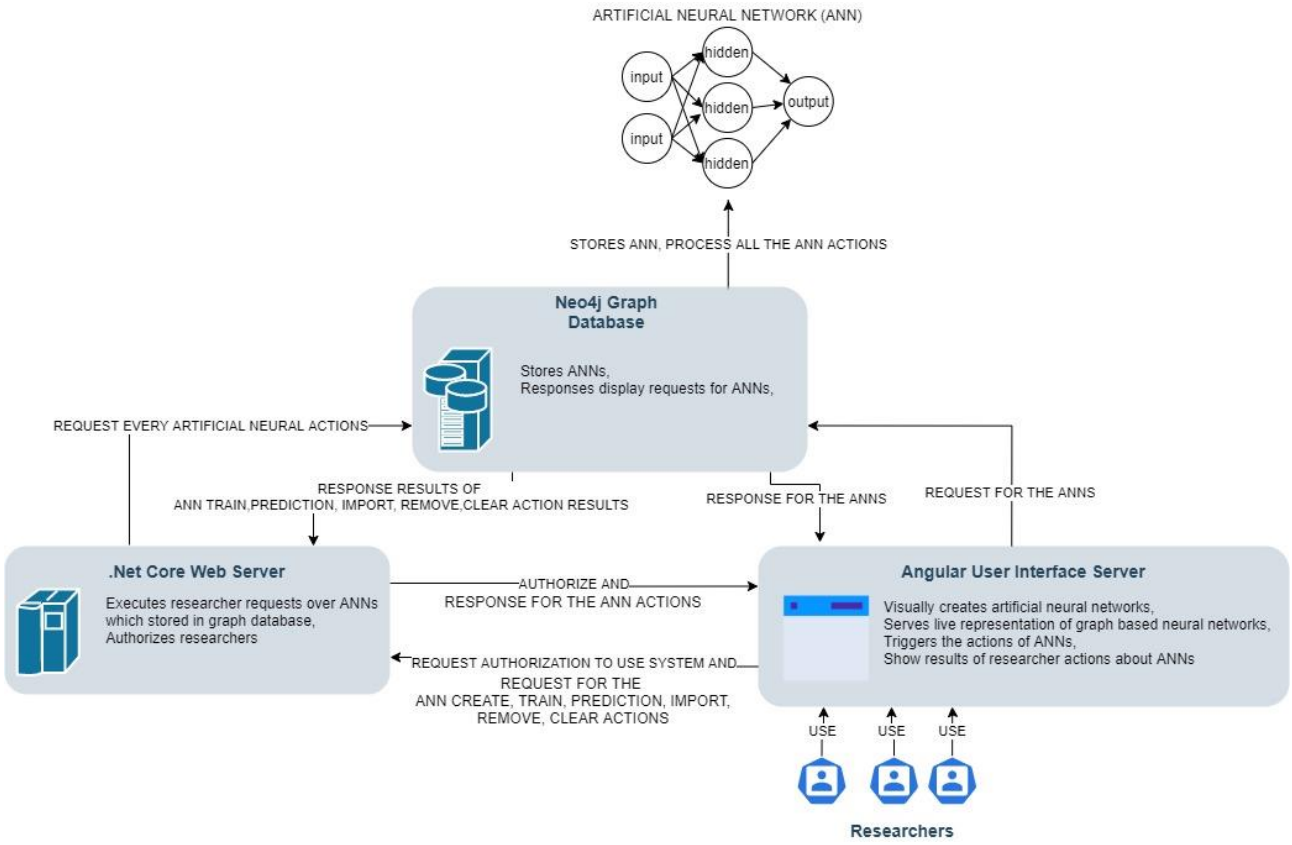


Figure 1. Software system architecture diagram shows layers of software and its relations

The software designed in the presented method works with the individual responsibility of each main part. For this purpose dynamic user interactions, backend service and graph database can work standalone. It makes the developed framework scalable. Designed system optimized to distribute heavy works, such as frontend service users' browsers is responsible for rendering large neural network models on graphs. Visually created ANN models' cypher queries are the responsibility of frontend service. Backend service is responsible to serve queries to the graph database. Also, the backend is responsible for the authentication of users. Socket communication used to dynamically serve data that is responsible for all three layers of the system.

To express, representation power of ANN models on graphs, a popular neural network Keras framework is selected. Trained Keras models are imported into the system. Also, an approach is developed to import Keras models to the system. Keras models are trained and created to show design approach. In Keras, researchers can access layers, not on the nodes. In developed model researchers that import their models to a system, can see every node on model and parameters also can see the layers if they want abstraction. Visualization of ANN models is the same one to one match as to how they are stored in the graph database. The study aims to make ANN modelling one to one representation and storing it like that. The model creation process is also.

Two different ways of representation are explained to do such storing possible. The first way is every matrix cell in multi-dimensional matrices of models are expressed as nodes. The second way is every matrix should be expressed as a node or relationship of nodes. In the study, the ways to be followed while realizing artificial neural networks in the Neo4j graph database were examined. There are also two different methods followed in Figure 12 and Figure 14.

Complex artificial neural network model results of the Keras (2017) realized in the study and transferred to graph database with the developed approach. Training of the artificial neural network model created and demonstration of the training results was realized. Technical problems encountered while creating, testing, visually observing, transferring popular ANN models are explained in the study. The estimation result of the handwritten number recognition model, which is an example problem of the convolutional neural network, in the developed system was compared with Keras. The study showed that the method developed works faster than the estimation made in Keras. A solution to problems is explained, also. The developed approach and system are shared as open-source (Graph Based Neural Network 2019).

2.2. Creating an artificial neural network model on graph database

In this section, the creation of artificial neural networks on graph databases using the proposed approach is explained. The actions that the researcher can take and the ANN creation process steps were examined. After the user logs in to the system from the home page, by selecting the "Create New Model" option on the left menu, they are directed to the model creation page and visually creates the artificial neural network model. Login section is shown in Figure 2. As an example, a NAND logic gate learner ANN is created on the system and the main screen is shown in Figure 3.

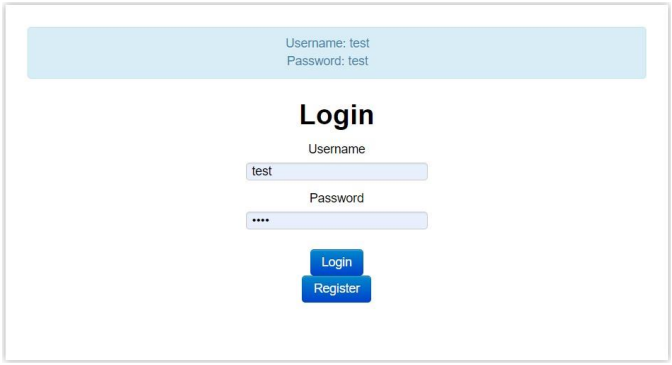


Figure 2. Researcher login screen

In Figure 3, an artificial neural network model is created to learn the NAND logic gate with a three-input model as an example over the system.

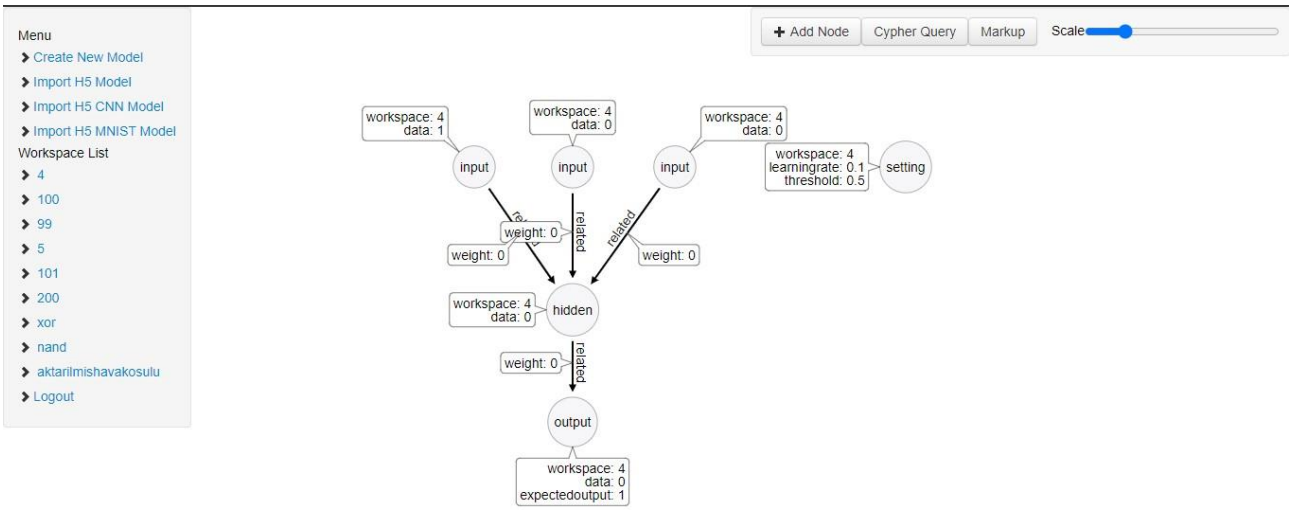


Figure 3. Visually created NAND feed-forward perceptron model

When the “Add Node” button is pressed on the interface and a node is created. To add a new node the "Add Node" button is clicked, the new nodes are displayed on the screen, and the nodes are connected by hovering over the node and dragging and dropping it on the other node. The researcher can design the entire artificial neural network by dragging and dropping with the cursor on the website, double-clicking on the node or relationship and clicking the "Add Node" button. When the created node is double-clicked, the “Edit Node” screen appears as shown in Figure 4. “Edit Node” screen is the editor for node data. A created node corresponds to the node in the graph database and also expresses a neuron in the neural network. The label must be entered to indicate that the neuron is in the input, hidden, or output layer. By writing the model's workspace, it tells the model which name will be recognized when testing and training follow-ups by the researchers. The data property indicates what data the neuron contains in its initial state.

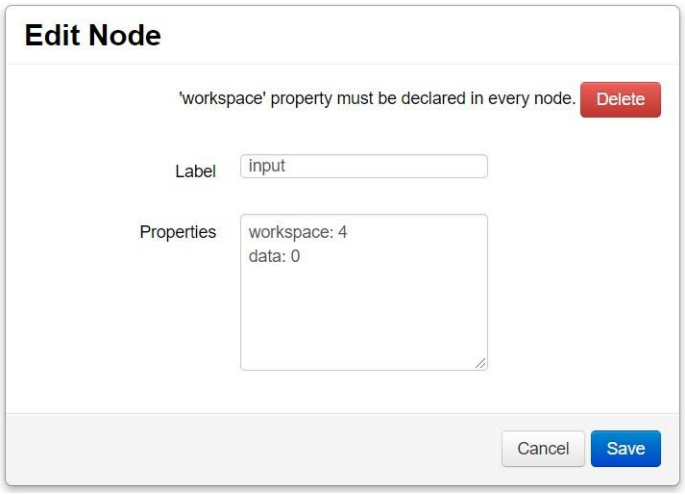


Figure 4. Node editing screen

At the same time, neurons can be dynamically created and linked, and editing screens can be opened and deleted. Relationships in the graph database are created as the equivalent of synapses to connect the neurons after the interface is created, as shown in Figure 5a.

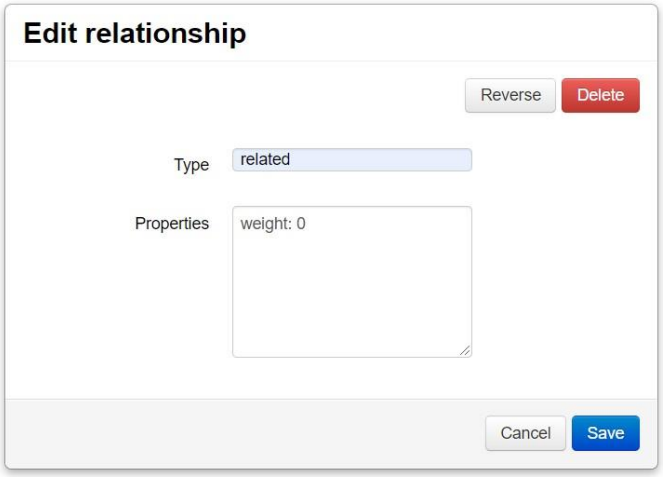


Figure 5a. Relationship editing screen

The type of relationship is used for naming. The "weight" value written in the property of the relationship corresponds to kernel, bias or weight values according to the type of model to be created. During model

training, when switching between neurons, multiply the properties in the relationship, sum, etc. transactions are made. In addition to the nodes and relationships, the setting node appears on the site, this node was created to hold the settings to be used while training, the node is displayed in Figure 3. In the model that learns the NAND gate with three inputs, the "learningrate" value expressed as the step interval is selected as 0.1, the "threshold" value of activation is selected as 0.5 and can be displayed in Figure 3. After the researcher creates the artificial neural network model, it can make it easier to display the model on the screen by changing the scale of the model. The visually created model is ready to be transferred to the graph database.

2.3. Transfering artificial neural network models into the graph database

After creating the artificial neural network, the researcher should display the screen with the model output written in the query language named Cypher for the transfer to the graph database. The output of the cypher query that will be run in the database of the model created on the site is displayed in Figure 5b. The data created in the query is kept to send neuron and synapse data to the graph database.

The query contains two different structures. After the "CREATE" command, neuron data is created with virtual id numbers, and then, while expressing relationships, these virtual id numbers connect neurons as they are created in the model. Using the open command on the console, how the query will be displayed in the graph database can be displayed n the new tab, and it is provided to register to the graph database with the "Create New Model" button. After the model is created, in the menu on the left, under the "Workspace List", a line with 4 is written for the model whose workspace is created and identified as 4. 4 under the workspace list represents a model environment where the status of the model can be monitored. Model registered in graph database via the system.

Before the three input NAND perceptron model diagram is imported into the database, the Cypher query is shown to the researcher. The query appears in Figure 5b. In Figure 6, the neural network created by the researcher on the system is shown in the Neo4j Browser. The query consists of two parts. The first of these sections is the node definitions, and the second is the relation definitions. From the lines used to create a node, for example, the leading 0 in the line "('0': input {workspace: '4', data: '0'})," indicates that the first node will be created virtually. Tags are given to the nodes in the query, for example ": input" plays a role in indicating that the node is an input node. The workspace variable was created to understand the fact that it will be used in the training and test sections, which are given to the node when creating the model, and to provide ease of query. The data is used to express the data to be stored in the node. After six nodes are defined in the query, relationship definitions are made. When defining a relationship in the query for example, the line "("0 ') - [:' related '{weight:' 0.0 '}] -> (' 1 ')," has a relationship between which node and which node. The direction of the relationship expressed with "->" arrow. The label of the relationship. and its weight has been defined in the weight field. In the example given in Figure 5b, in the query that creates the relationship there is a command to establish a relation between the zeroth node and first in the query between zero and one, with a weight of 0.0, with the relationship name called as related. For example without node definitions, the relations of the first input node without '0' cannot be identified in the query. "Create New Model" button should be clicked if the researcher wants to create a model in the graph database. After the researcher created the model, Figure 6 shows the ANN generated in the Neo4j database.

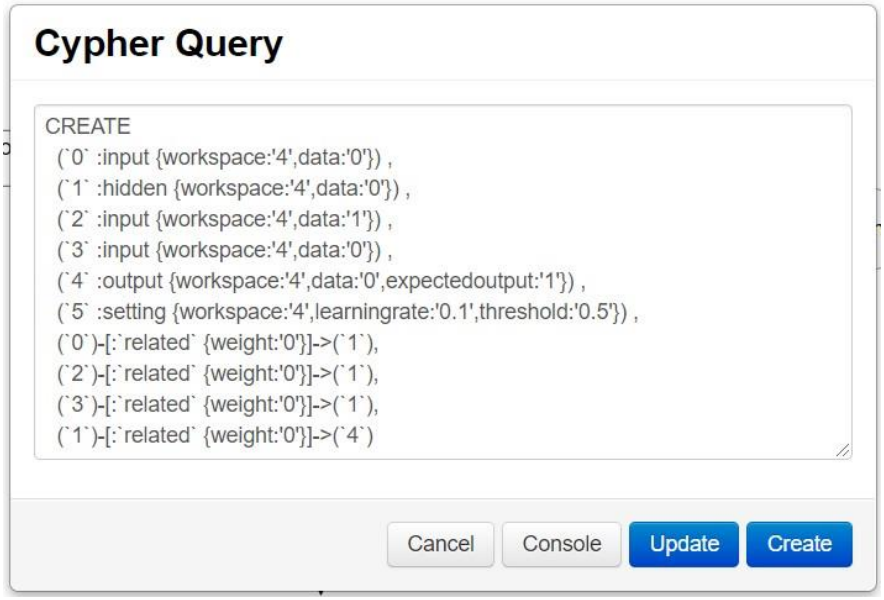


Figure 5b. NAND logic gate learner neural model Cypher query output screen

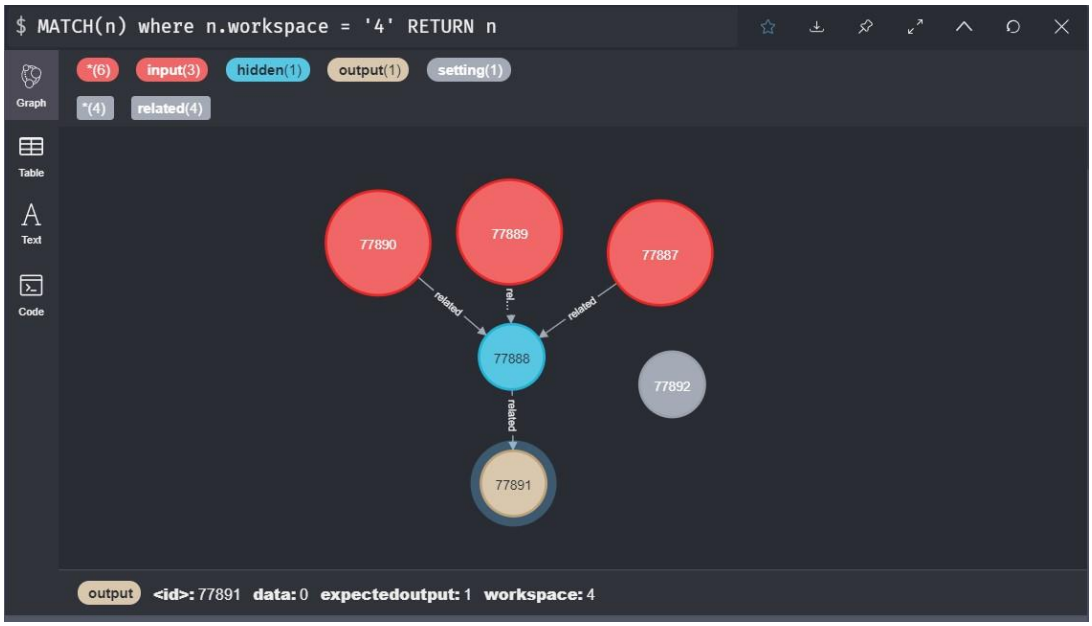


Figure 6. NAND logic gate learner ANN model representation on Neo4j browser

2.4. Transfering trained artificial neural models

A structure has been developed to transfer and visualize artificial neural network models and perform predictions on Tensorflow (2015) and Keras (2017) frameworks models. Models trained in Keras can be transferred to the developed system if they are exported with their configurations and weights in h5 file format.

An exemplary feed-forward neural network model for transferring the H5 format to the system was created on the Keras software framework. The model is trained to learn whether someone will play football depending on the weather, humidity, wind variables. Table 1 shows the data used in the feed-forward network model and whether or not football is played as a result. It is used in the ANN model if the weather condition is sunny, 0 if it is overcast and -1 if it is rainy. Moisture value is evaluated as 0 if normal and 1 if

high. It is evaluated as 0 if the wind value is weak and 1 if it is strong. If the value of playing the game is yes, it's evaluated as 1, if no it is evaluated as 0.

The sample data set is shown in Table 1. The feed-forward Keras ANN model which decided to play football was created with the sequential definition of Keras. Tanh is used as the activation function in the input layer of the model since it must receive inputs in the range from -1 to 1. In the output layer, the sigmoid activation function is used because the outputs must be 0 or 1. Mean squared error was used for stochastic gradient descent and loss or cost function for optimization. The model training stopped when it produces values close to the desired results after the five thousand epochs have been trained and took its final form. The h5 output of the model was taken under the name "demo_model.h5" which is provided in project codes.

Table 1
Data set of the feed-forward model

Weather condition	**Moisture**	**Wind**	**Play**
Sunny	High	Weak	No
Sunny	High	Strong	No
Overcast	High	Weak	Yes
Rainy	High	Weak	Yes
Rainy	Normal	Weak	Yes
Rainy	Normal	Strong	No
Overcast	Normal	Strong	Yes
Sunny	High	Weak	No
Sunny	Normal	Weak	Yes
Rainy	Normal	Weak	Yes
Sunny	Normal	Strong	Yes
Overcast	High	Strong	Yes
Overcast	Normal	Weak	Yes
Rainy	High	Strong	No

Using the "demo_model.h5" example model in framework content files, with the "Import H5 Model" option on the site, firstly the "h5tojson.py" python code is converted from the h5 model to the JSON data format where the layers and neurons are easily processed. Lastly, the JSON model data converted to the graph database query with the developed framework. An example of an artificial neural network converted to a query can be seen in Figure 3. The status of the model mentioned in the study after the transfer from h5 format to the study project is shown in Figure 7. The model has been successfully transferred to show the three weather inputs via data, kernel, bias values in the feature fields on their relationship, and to show in the output node as the output whether it will play football. In Figure 7, data in all relationships and nodes are not shown, only relationships in one path are shown.

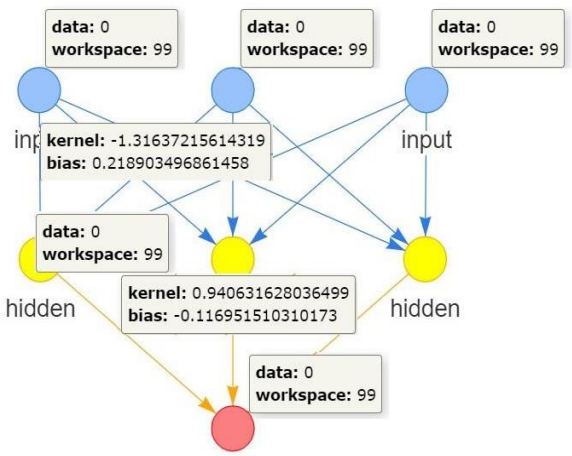


Figure 7. Representation of the transferred feed-forward h5 model

By transferring and visualizing the model into the study, which input node has the highest priority, its working structure can be observed more clearly and it enables us to watch the training and test process live with the socket connection. By transferring the model to the system and running the graph database on the server, it is prevented to have H5 output to move the model as in Keras. The view of the feed-forward model completed and transferred over Neo4j is as in Figure 8, all data and relations are made and kept on the graph database. Thus the transferred ANN which is now in the graph database can be used by researchers as desired and can be accessed online. ANN is implemented in an ecosystem that will not need the cycle again for transmission. While the model cannot be monitored visually while in Keras, all processes can be monitored visually with the proposed system. While the researchers cannot work together on the model with the method offered by Keras, they can perform collaborative studies with the method presented.

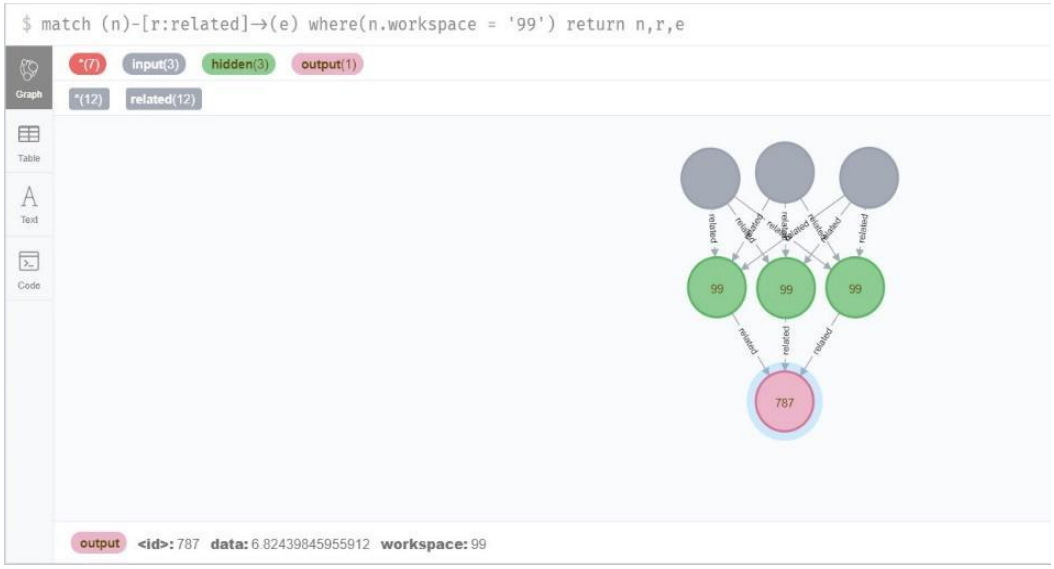


Figure 8. Query representation of the transferred h5 model in Neo4j

2.5. Testing the artificial neural network model on the graph database

In the widely used software frameworks such as Keras and Tensorflow, researchers observe model training and testing courses with an error rate, learning rate, etc. Model creation and testing processes are monitored from the parameters and the inference that if the model trained enough or not. While training or testing, researchers have no chance to view the visual status of the model. To eliminate this problem, the test parameters were entered into artificial neural network models on the Neo4j graph database and the status of the test was queried live with its socket communication structure. In Figure 9, a researcher is watching the test process live. By clicking on the workspace name on the left menu, the screen where the models are displayed. In the upper right menu, there is a stop button for ending live tracking, the screen where the test parameters can be entered, and the training button where the training can be started.

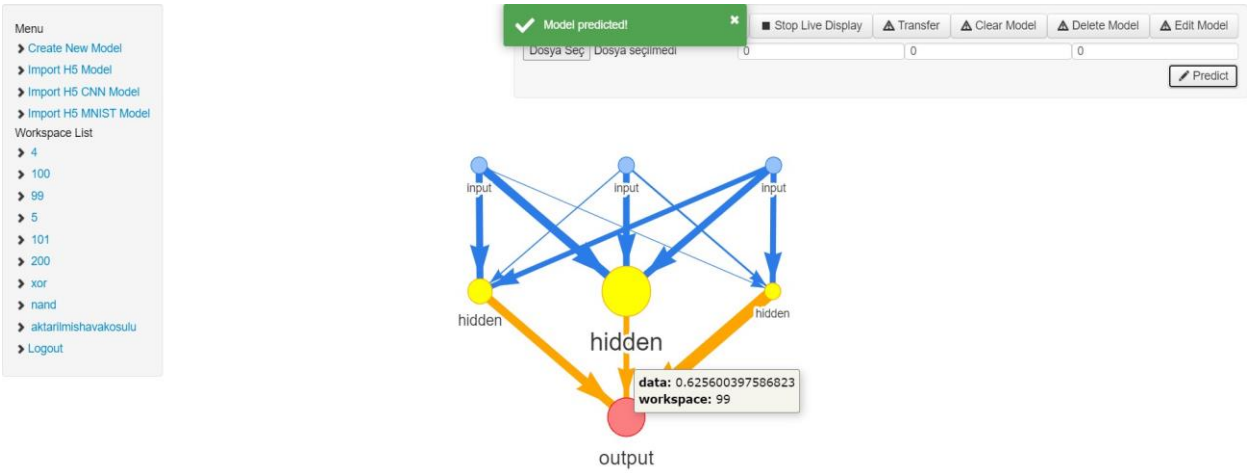


Figure 9. The screen of the researcher conducting the prediction in the feed-forward model

When the parameter input is tested with sample data such as overcast weather, normal humidity, weak wind, i.e. 0,0,0 inputs, the prediction results are calculated and stored in the database, as shown in Figure 9. In Figure 9, researcher trigger prediction process of displayed ANN. Figure 9 shows that the ANN model predicts play football with 62% probability. Another researcher can display test process at the same time and can see the prediction results. Between hidden and input layers Tanh activation function used. Between output and hidden layers, Sigmoid activation function is used.

The figure shows the current status of neurons or relationships over the site. Also, the size of the nodes refers to the size of the data. If the output data is between 0.5 and 1 it means that football will be played, if its between 0.5 and 0 football will not be played. In this case, the result of the test indicates that football will be played. It matches according to the test result and dataset. While performing the test, the data in the node is obtained by multiplying the kernel of the relationship with the previous node and adding bias on it. This continues in the same way until it reaches the output and ends when the output is achieved.

In this study, a NAND logic gate learner perceptron with three inputs was created to perform the training. As seen in Figure 10, input neurons are given [1,0,0]. The value of 1 was given as expected output. Activation will work at values above 0.5 and weights will be increased by increasing 0.1 steps.

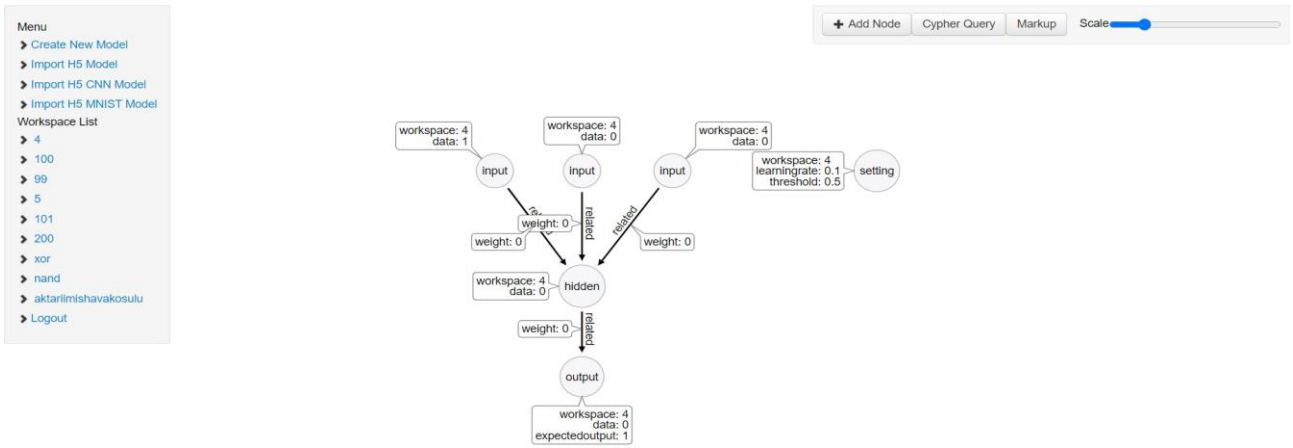


Figure 10. The image after the creation of a model that learns the NAND logic gate created by the researcher on the system.

While performing the training in Figure 11, the process can be observed live, and the result is presented to the researcher in green notification popup. The expected result was obtained in two steps and model weights were assigned. Relationship weights that increase as much as the value of learning rate in each step for learning continue to increase if not equal to expected output. All the processes in the estimation and learning stages were made in the graph database using graph theories. In the perceptron learning performed in Figure 11, the value in each input neuron was transferred by adding the weight in the relationship while passing to the next neuron. When the data value in neurons exceeds the threshold value, the activation is triggered and the outputs are found and the expected output is checked for equality. As can be seen in Figure 11, learning was stopped by providing equality to expected output in 2 steps.

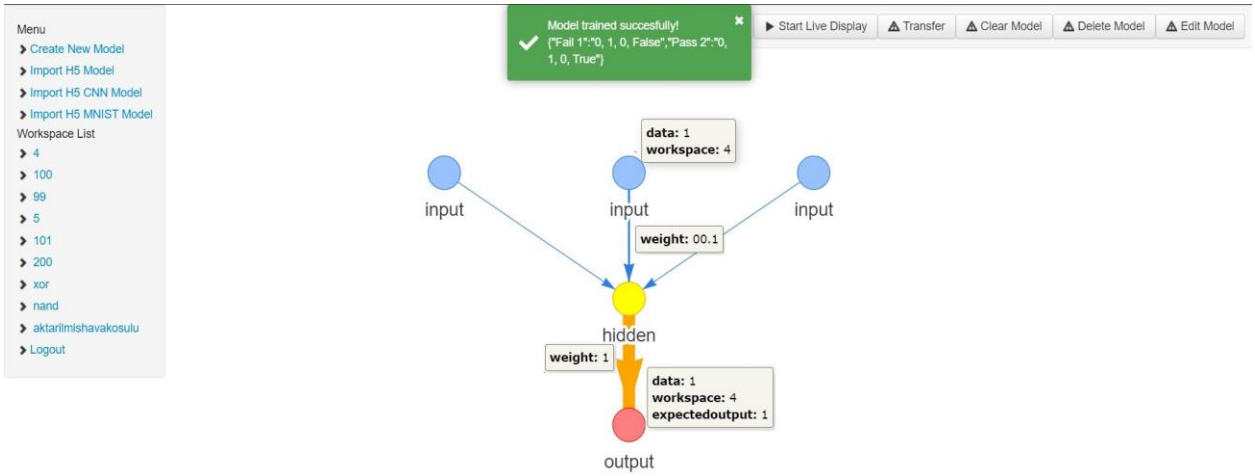


Figure 11. The panel where artificial neural network training is observed

The developed framework was also made to strengthen the representation of convolutional neural network models. The imported convolutional neural network model keeps all its data and structure in the graph database and allows it to be processed and observed. Figure 12 shows a part of the CNN model stored in the system. MobileNet version 2 (Howard et al., 2017) is used to show the import of convolutional neural models into the system. Model matrices are stored as every single matrix as a node. The way is not efficient to store big blob data in node properties as the Neo4j graph database framework suggests. Layer representation is developed to simplify visualization on large models. CNN model layer representation is

shown in Figure 13. The way of model representation on the browser can be zoomed in or out. Also, drag actions can be done on nodes, to observe better. WebGL technology has been used to facilitate the render process in the browser in networks containing many neurons and layers such as CNN.

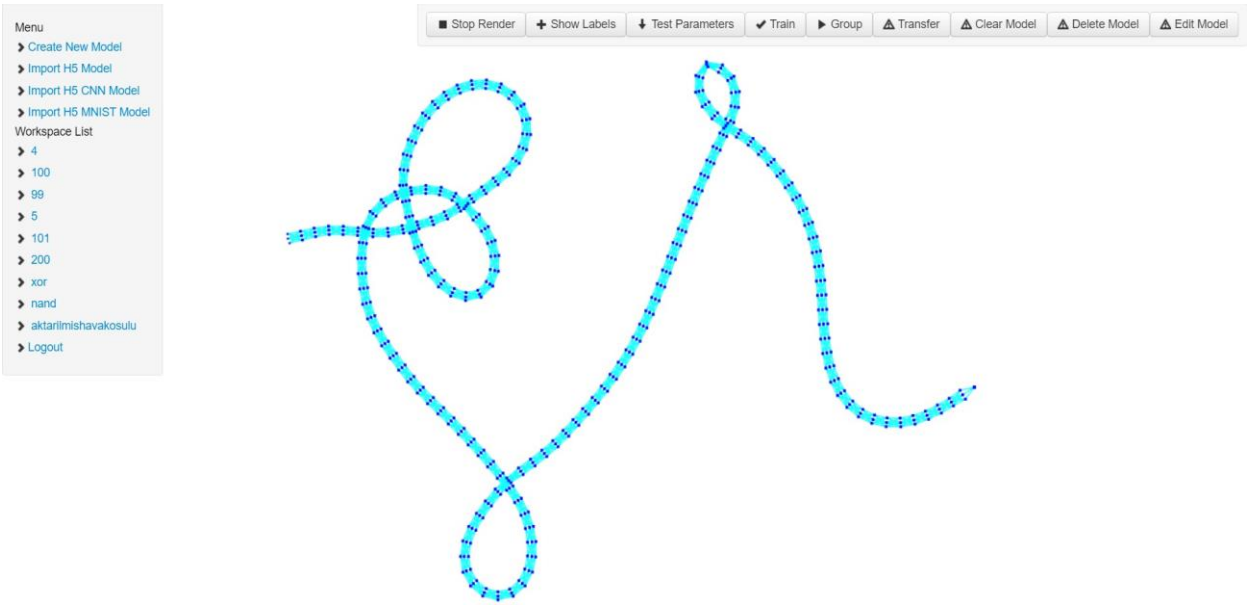


Figure 12. Imported MobileNet convolutional neural network representation

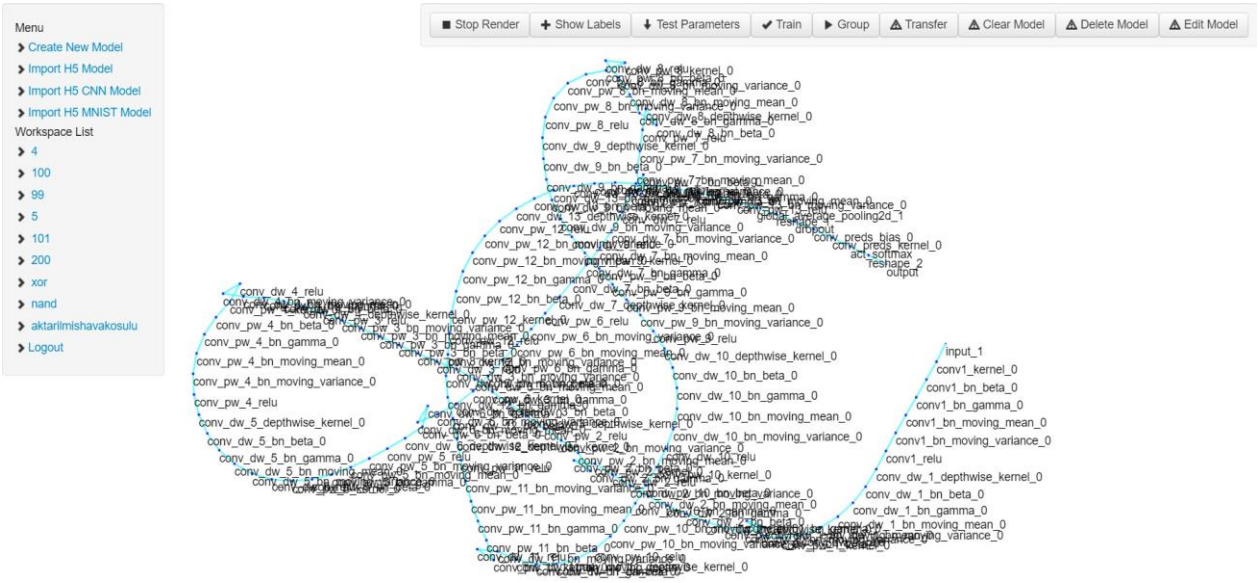


Figure 13. Layer representation of MobileNet network

Well known handwritten digit dataset MNIST (LeCun, Cortes & Burges 2010) trained with Keras framework and imported to the system from hd5 file format of the trained model. The model predicts 28x28 handwritten digits. MNIST data set was a performance criterion of early computer vision machine learning times. Since the MNIST data set is an important toy case, it was preferred in this study. With the importing of the model, every single cell in matrices represents a node to show different ways of holding data in the graph database. The model is shown in Figure 14 with every node represents a cell in the kernel, bias matrices and their output matrices. In Figure 14, nodes in the graph stores matrix cell data on properties. Relationships between

nodes represent connections of nodes and these connections can be virtualized as layer nodes as seen in Figure 15. In Figure 15, the grouping feature of the system executed then the same label nodes of ANN shown as layers to the researchers. Any kind of parameter of ANNs in the system which node or relationship has can be shown with node representation on the web site in case the researcher wants to see. Researchers see the model life cycle and how it's stored and processed. Nodes standing apart on the left represent the input and first matrix multiplier, respectively. Large nodes on the of the nodes forming the ring represent the output of the bias layer. The right side of the nodes forming the ring represents the output of the kernel layer. Nodes that form the ring shape are holding big matrices cell data on nodes. While testing the model every node is used to calculate outputs.

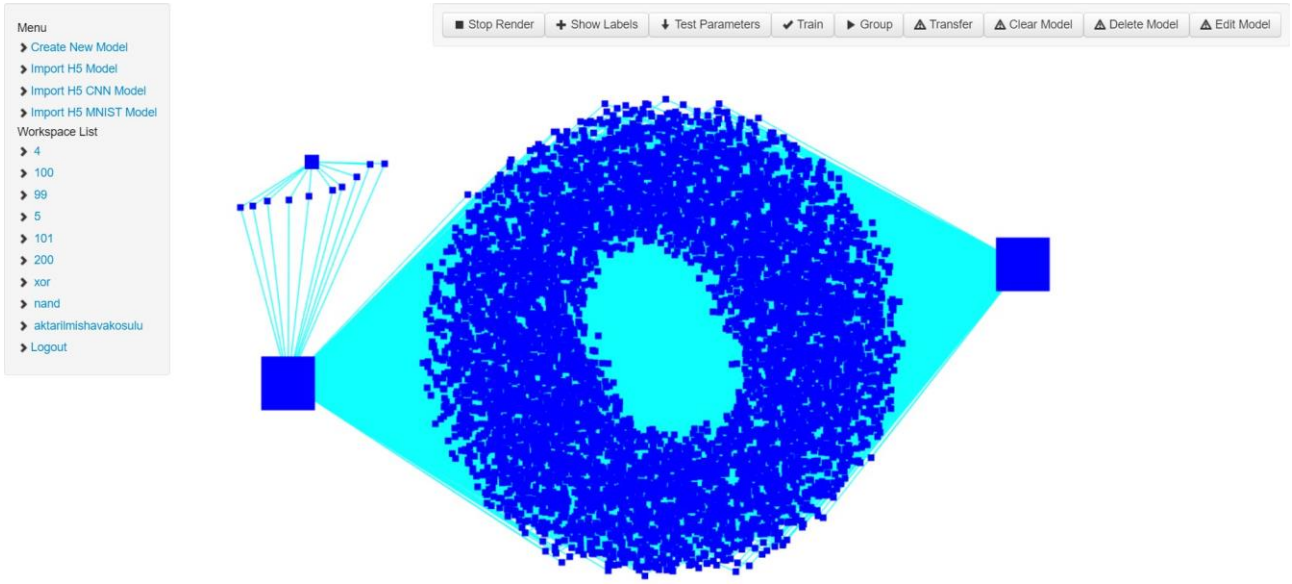


Figure 14. Imported MNIST handwritten digit dataset trained ANN model display in graph form on web site

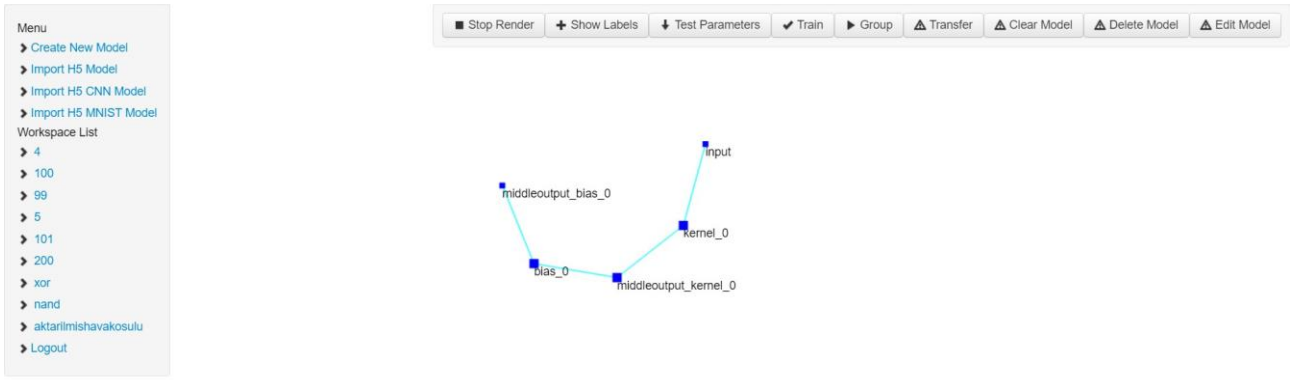


Figure 15. Imported MNIST handwritten digit dataset trained ANN model display on web site with only layer representations

Neo4j does not support multi-dimensional matrices in property values by default. For this reason, in the model described Figure 12 the artificial neural network is represented exactly, but the calculation costs increase. On the other hand, in the Mnist model shown in Figure 14, each cell of the matrices in the layers was expressed as a neuron. Thus, Neo4j directly supported the model data structure used. Although it makes the observation power of the model difficult, it reduces the computational costs. Both methods were

implemented in the proposed system and the difference was examined in this way. In the next section, a sample prediction is made and a comparison of Keras and performance is made. In the next section, a sample prediction is made and performance comparison of Keras and the proposed system is made.

2.6. MNIST handwritten digit artificial neural network model prediction comparison

With the Tensorflow 2.0, Keras is officially used to perform high-level operations. Keras is widely accepted by researchers and industry for deep learning and convolutional neural network focused work. In this section, the model prediction has been performed with both Keras and the proposed graph database based neural network study. For comparison, the 2.3.0 version of Keras and the Tensorflow library that supports graphics cards are used. Using the version of Tensorflow that performs calculations on the GPU, the library with the highest performance has been selected for comparison. Comparisons were made on a laptop with Intel i7-7700HQ processor, 16 GB DDR4 memory and Nvidia 1050Ti graphics card.

The Mnist dataset includes 28 by 28-pixel hand-drawn figure images. In this data set, the learning was done on the convolutional neural network. ReLU activation function is used in convolution layers and SoftMax activation function in the output layer. Grayscale visual matrices of the picture take values between 0 and 255. The number drawing with the prediction test is shown in Figure 16. While the estimation is performed in the proposed system, the researcher is expected to upload the image to the system. After selecting the file, the image is transformed into a 1 by 784 matrix with the code running in the browser of the researcher who starts the prediction. Converting the image to a matrix will not incur computational costs on the backend server. As a result of multiplication and sum operations performed with calculation matrices, a 1 by 10 matrix is obtained and the matrix index containing the highest number shows the estimated number. After the prediction is concluded, the researcher is shown the estimated number in the green popup. The researcher can see the data in the neural network layers on the site, as well as the results of each neuron.



Figure 16. 28 by 28 pixel hand-drawn number 9 to be predicted in the Mnist model

In every attempt made with Keras, there is a cost of distributing the input and the model to the GPU, as all computing is done on the GPU. The data needs to be transformed every time. This can cause slow running. As the Neo4j graph database runs, it caches and returns queries quickly. Therefore, in the graph database based neural network the initial slowness seen in Table 2 is gradually decreasing. In Table 2, the return of model estimation results in Keras and the proposed system is given in seconds. The proposed system can give 125 milliseconds faster estimation output on the arithmetic average than Keras based on the graphics card. Because the proposed system is online, network status, server density can change the calculation times. The ability of the proposed system to calculate faster than the accepted libraries shows the power of graph database-based artificial neural networks. All calculation results are kept in artificial neural network cells and links in the graph database.

Table 2
Comparison of the recommended system and GPU supported Keras prediction times

Trial	**Graph database based neural network (sec)**	**Keras (sec)**
1	3.34	3.41
2	3.37	3.18
3	3.34	3.15
4	3.28	2.97
5	3.31	3.19
6	3.32	3.19
7	3.26	3.76
8	3.29	3.18
9	3.66	3.09
10	3.36	3.16
Arithmetic Mean	3.353	3.228

As can be inferred from the comparison, although the main focus of this new study in the field is to express artificial neural networks better and enable the development of new types of networks, its performance is also acceptable.

3. Conclusion

In this study, a completely new method has been developed and artificial neural networks are processed in graph databases. With this study, artificial neural networks implemented in graph databases can be designed visually by researchers. Researchers can edit the artificial neural network graph shown on the developed software without the need for a code. It is ensured that researchers can operate processes on the same ANN models together. With its infrastructure, researchers can share artificial neural network models without transforming. The one-to-one representation of artificial neural networks in the theory was realized in the graph database. The black box problem is illuminated by observing every element and flow in artificial neural networks kept in the graph database. By comparing the estimation with Keras, it is concluded that the calculation speed of the proposed system is acceptable and makes predictions faster in the example. Since it is shared with open-source code, the first steps of the graph database-based neural network ecosystem have been taken.

In subsequent studies, apart from the artificial neural networks suggested in the study, it can be provided to operate different artificial neural networks. Generalizable prediction and training phases can be designed for graph database-based neural networks. Performance optimizations for training and testing of convolutional neural network models for performing transactions on the graphics card can be developed. In industrial projects, it can be proved that the presented method can work at industry standards by realizing artificial neural networks where data is kept in graph databases.

Acknowledgement

This work was supported by the Hepsiburada. I would like to thank Hepsiburada for the opportunities it has given.

Author Contributions

Doğa Barış ÖZDEMİR: Developed the software platform, generated models for proof of concept.

Ahmet Cumhur KINACI: Designed and analyzed the artificial neural network processes on graphs approach.

Conflicts of Interest

The authors declare no conflict of interest.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. Cognitive science, 9(1), 147-169.

Armenta, M. A., & Jodoin, P. M. (2020). The Representation Theory of Neural Networks. arXiv preprint arXiv:2007.12213.

Battaglia, W., P., Hamrick, B., J., Bapst, Alvaro, … Razvan. (2018, October 17). Relational inductive biases, deep learning, and graph networks. Retrieved from https://arxiv.org/abs/1806.01261.

Buhrmester, V., Münch, D., & Arens, M. (2019). Analysis of explainers of black box deep neural networks for computer vision: A survey. arXiv preprint arXiv:1911.12116.

Carpenter, G. A., & Grossberg, S. (1990). ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. Neural networks, 3(2), 129-152.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In Artificial intelligence and statistics (pp. 192-204).

Cvitkovic, M. (2020). Supervised Learning on Relational Databases with Graph Neural Networks. arXiv preprint arXiv:2002.02046.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.

Çuhadar, M., & Kayacan, C. (2005). Yapay Sinir Ağları Kullanılarak Konaklama İşletmelerinde Doluluk Oranı Tahmini: Türkiye'deki Konaklama İşletmeleri Üzerine Bir Deneme. Anatolia: Turizm Arastirmalari Dergisi, 16(1).

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae, 128-140. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial networks. arXiv preprint arXiv:1406.2661, 4(5), 6.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.

Graph Based Neural Network (2019). Retrieved from https://github.com/dogabaris/GraphBasedNeuralNetwork.

Hebb, D. O. (1949). The organization of behavior: a neuropsychological theory. J. Wiley; Chapman & Hall.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554-2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the national academy of sciences, 81(10), 3088-3092.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Keras: The Python Deep Learning library. (2017). Retrieved from https://keras.io.

Kohonen, T. (1982). Self-organized Formation of Topologically Correct Feature Maps. Biological Cybernetics, 43, 59-69.

Kohonen, T. (1990). The self-organizing map. Proceedings of the IEEE, 78(9), 1464-1480.

Lam, H. T., Minh, T. N., Sinn, M., Buesser, B., & Wistuba, M. (2018). Neural feature learning from relational database. arXiv preprint arXiv:1801.05372.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database.

Liu, H. (2017, November 1). Hierarchical Representations for Efficient Architecture Search. Retrieved from https://arxiv.org/abs/1711.00436v2.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013, June). Rectifier nonlinearities improve neural network acoustic models. In Proc. icml (Vol. 30, No. 1, p. 3).

Mcculloch, W. & Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5, 127--147.

Minsky, M., Papert, S. (1969). Perceptrons: An Introduction to Computational Geometry. Cambridge, MA, USA: MIT Press.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. Neural computation, 1(2), 281-294.

Muhammad, T., & Halim, Z. (2016). Employing artificial neural networks for constructing metadata-based model to automatically select an appropriate data visualization technique. Applied Soft Computing, 49, 365–384. DOI: 10.1016/j.asoc.2016.08.039.

Nair, V., & Hinton, G. E. (2010, January). Rectified linear units improve restricted boltzmann machines. In ICML.

Nekhaev, D., & Demin, V. (2017). Visualization of maximizing images with deconvolutional optimization method for neurons in deep neural networks. Procedia Computer Science, 119, 174–181. DOI: 10.1016/j.procs.2017.11.174.

Neo4j. (2007). Retrieved from https://neo4j.com.

Olden, J. D., & Jackson, D. A. (2002). Illuminating the “black box”: a randomization approach for understanding variable contributions in artificial neural networks. Ecological Modelling, 154(1-2), 135–150. DOI: 10.1016/s0304-3800(02)00064-9.

Öztanır, O. (2018). Makine Öğrenmesi Kullanılarak Kestirimci Bakım (Master's thesis, Fen Bilimleri Enstitüsü).

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408. https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1985). Learning Internal Representations by Error Propagation. In D. E. Rumelhart & J. L. Mcclelland (ed.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations (pp. 318--362). MIT Press.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. IEEE Transactions on Neural Networks, 20(1), 61-80.

Schikuta, E. (2008). Neural networks and database systems. arXiv preprint arXiv:0802.3582.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Colorado Univ at Boulder Dept of Computer Science.

Tank, D., & Hopfield, J. (1986). Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. IEEE transactions on circuits and systems, 33(5), 533-541.

TensorFlow. (2015). Retrieved from https://tensorflow.org.

Tosun, S. (2007). Sınıflandırmada yapay sinir ağları ve karar ağaçları karşılaştırması: Öğrenci başarıları üzerine bir uygulama (Doctoral dissertation, Fen Bilimleri Enstitüsü).

Touretzky, D. S., & Pomerleau, D. A. (1989). What's hidden in the hidden layers. Byte, 14(8), 227-233.

Uwents, W., Monfardini, G., Blockeel, H., Gori, M., & Scarselli, F. (2010). Neural networks for relational learning: an experimental comparison. Machine Learning, 82(3), 315–349. DOI: 10.1007/s10994-010-5196-5.

Wang, T. (2018, February 15). NerveNet: Learning Structured Policy with Graph Neural Networks. Retrieved from https://openreview.net/forum?id=S1sqHMZCb

Widrow, B. & Hoff, M. E. (1960). Adaptive Switching Circuits. 1960 IRE WESCON Convention Record, Part 4 (p./pp. 96--104), New York: IRE.

Widrow, B., & Lehr, M. A. (1990). 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proceedings of the IEEE, 78(9), 1415-1442.

Witt, C., Bux, M., Gusew, W., & Leser, U. (2019). Predictive performance modeling for distributed batch processing using black box monitoring and machine learning. Information Systems, 82, 33–52. DOI: 10.1016/j.is.2019.01.006.

Yahia, M. E., & Elsawi, A. M. (2003). Neural Database Model.