ТЕХНИЧЕСКИЕ НАУКИ
И ТЕХНОЛОГИИ

АҚПАРАТТЫҚ-КОММУНИКАЦИЯЛЫҚ ТЕХНОЛОГИЯЛАР
ИНФОРМАЦИОННЫЕ И КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ
INFORMATION AND COMMUNICATION TECHNOLOGIES

DOI 10.51885/1561-4212_2024_1_190
МРНТИ 20.01.07

K. Maksutova[1], N. Saparkhojayev[2], Dusmat Zhamangarin[3]
[1]L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
*E-mail: qunkabai@gmail.com**
[2]D. Serikbayev East Kazakhstan technical university, Ust-Kamenogorsk, Kazakhstan
E-mail: n.saparkhojayev@sci.gov.kz
[3]Kazakh Technology and business University, Astana, Kazakhstan
E-mail: Dus_man89@mail.ru

DEVELOPMENT OF AN ONTOLOGICAL MODEL OF DEEP LEARNING NEURAL NETWORKS

ТЕРЕҢ ОҚЫТУДЫҢ НЕЙРОНДЫҚ ЖЕЛІЛЕРІНІҢ МАТЕМАТИКАЛЫҚ МОДЕЛІНІҢ АЛГОРИТМІН ЖАСАУ. ОНТОЛОГИЯЛЫҚ МОДЕЛЬДІ ӘЗІРЛЕУ

РАЗРАБОТКА АЛГОРИТМА МАТЕМАТИЧЕСКОЙ МОДЕЛИ НЕЙРОННЫХ СЕТЕЙ ГЛУБОКОГО ОБУЧЕНИЯ. РАЗРАБОТКА ОНТОЛОГИЧЕСКОЙ МОДЕЛИ

Abstract. *This research paper examines the challenges and prospects associated with the integration of artificial neural networks and knowledge bases. The focus is on leveraging this integration to address practical problems. The paper explores the development, training, and integration of artificial neural networks, emphasizing their adaptation to knowledge bases. This adaptation involves processes such as integration, communication, representation of ontological structures, and interpretation by the knowledge base of the artificial neural network's representation through input and output.*

The paper also delves into the direction of establishing an intellectual environment conducive to the development, training, and integration of adapted artificial neural networks with knowledge bases. The knowledge base embedded in an artificial neural network is constructed using a homogeneous semantic network, and knowledge processing employs a multi-agent approach.

The representation of artificial neural networks and their specifications within a unified semantic model of knowledge representation is detailed, encompassing text-based specifications in the language of knowledge representation with theoretical semantics. The models shared with the knowledge base include dynamic and other types that vary in their capabilities for knowledge representation.

Furthermore, the paper conducts an analysis of approaches to creating artificial neural networks across various libraries of the high-level programming language Python. It explores techniques for developing artificial neural networks within the Python development environment, investigating the key features and functions of these libraries. A comparative analysis of neural networks created in object-oriented programming languages is provided, along with the development of an ontological model for deep learning neural networks.

Keywords: *model, neuron, mathematics, neural networks, knowledge bases, ontological model.*

Аңдатпа. *Бұл мақалада жасанды нейрондық желілерді білім қорымен интеграциялау мәселелері мен бағыттары сипатталған. Қолданбалы есептерді шешу үшін білім базасы мен жасанды нейрондық желінің интеграциясын пайдалану үшін жасанды нейрондық желіні әзірлеу, оқыту және интеграциялау интеграция, байланыс, онтологиялық құрылымды ұсыну және жасанды нейрондық желінің кірісі мен шығысы арқылы жасанды нейрондық желінің көрінісін білім* *базасының интерпретациясы арқылы білім базасына бейімделеді. Бейімделген жасанды нейрондық желілерді білім қорымен дамыту, оқыту және біріктіру үшін Интеллектуалды ортаны құру бағыты қарастырылады. Жасанды нейрондық желіге енгізілген білім базасы біртекті семантикалық желі негізінде құрылады және ондағы білімді өңдеу мультиагенттік тәсіл арқылы жүзеге асырылады. Жасанды нейрондық желілерді ұсынудың онтологиялық моделі және олардың спецификациясы білімді ұсынудың бірыңғай семантикалық моделінде ұсынылған, ол білімді ұсыну тіліндегі мәтіндік түрдегі жасанды нейрондық желілердің спецификациясын теориялық семантикамен және білім базасына ортақ модельдермен, динамикалық және білімді ұсыну мүмкіндіктерімен ерекшеленетін басқа түрлермен қамтиды. Бұл жұмыста Python жоғары деңгейлі бағдарламалау тілінің әртүрлі кітапханаларында жасанды нейрондық желілерді құру тәсілдеріне талдау жасалды; Python әзірлеу ортасында жасанды нейрондық желілерді құру тәсілдері қарастырылады. Осы кітапханалардың негізгі ерекшеліктері мен функциялары зерттеледі. Объектіге бағытталған бағдарламалау тілдері тұрғысынан жасалған нейрондық желілерге салыстырмалы талдау жасалады. Терең оқытудың нейрондық желілерінің онтологиялық моделі жасалды.*

Түйін сөздер: *модель, нейрон, математика, нейрондық желілер, білім базалары, онтологиялық модель.*

Аннотация. *В данной статье рассматриваются трудности и направления интеграции искусственных нейронных сетей с базами знаний. Для решения прикладных задач используется интеграция базы знаний и искусственной нейронной сети, при этом разработка, обучение и интеграция искусственной нейронной сети адаптируются к базе знаний через интеграцию, связь, представление онтологической структуры и интерпретацию представления искусственной нейронной сети через входные и выходные данные. Рассматривается создание интеллектуальной среды для разработки, обучения и интеграции адаптированных искусственных нейронных сетей с базами знаний.*

База знаний, встроенная в искусственную нейронную сеть, основывается на однородной семантической сети, а обработка знаний в ней осуществляется с использованием мультиагентного подхода. Представлена онтологическая модель искусственных нейронных сетей и их спецификации в единой семантической модели представления знаний. Эта модель включает спецификацию искусственных нейронных сетей в текстовом виде на языке представления знаний с теоретической семантикой и моделями, общими для базы знаний, динамической и других типов, различающихся по возможностям представления знаний.

В работе проведен анализ подходов к созданию искусственных нейронных сетей в различных библиотеках языка программирования Python; рассмотрены методы создания искусственных нейронных сетей в среде разработки Python. Исследованы основные особенности и функции этих библиотек. Представлен сравнительный анализ нейронных сетей, созданных в терминах объектно-ориентированных языков программирования. Разработана онтологическая модель нейронных сетей глубокого обучения.

Ключевые слова: *модель, нейрон, математика, нейронные сети, базы знаний, онтологическая модель.*

Introduction. In 1943, W. McCulloch, an American scientist, and his student Walter Pitts formulated the inaugural mathematical representation of a neuron, the fundamental unit of the brain. Their work also established a foundational theory about brain functioning. Their impact can be summarized in three main areas:

1. They developed a foundational neuron model, acting as the basic processing unit. This model calculated the transition function by assessing the scalar product of the input signal vector and the vector of weight coefficients.
2. They introduced a network structure incorporating these neurons, enabling the execution of both logical and arithmetic operations.
3. Their fundamental assertion was that these networks had the capacity to learn, recognize patterns, and generalize acquired information.

In the realm of artificial intelligence (AI) and machine learning, the evolution of deep learning neural networks stands as a cornerstone of innovation. These intricate systems, inspired by the biological structure of the human brain, have demonstrated remarkable capabilities in tasks ranging from image recognition and natural language processing to autonomous driving and medical diagnosis. However, as deep learning continues to proliferate across diverse domains, the need for a systematic and comprehensive understanding of its underlying principles becomes increasingly paramount.

The development of an ontological model of deep learning neural networks emerges as a pivotal endeavor in addressing this imperative. By structuring and formalizing the knowledge inherent in these complex systems, an ontological model provides a unified framework for conceptualizing, organizing, and navigating the multifaceted landscape of deep learning. This endeavor transcends mere documentation; it embodies a systematic approach to elucidating the intricate relationships among the myriad components, architectures, algorithms, and methodologies that constitute deep learning neural networks.

The relevance of such a venture permeates multiple dimensions of contemporary scientific and technological discourse. Firstly, within the academic sphere, an ontological model serves as a catalyst for interdisciplinary integration, fostering collaboration among researchers from diverse domains such as computer science, mathematics, neuroscience, and cognitive science. It provides a common language and conceptual scaffold upon which disparate strands of knowledge can converge, thereby facilitating synergistic advancements in understanding and innovation.

Moreover, in practical applications, the utility of an ontological model extends far beyond academia, permeating industries and sectors where deep learning neural networks play a pivotal role. From healthcare and finance to autonomous vehicles and natural language processing, the adoption of deep learning technologies continues to proliferate, driving transformative shifts in how we perceive and interact with the world. Within these domains, an ontological model serves as a roadmap, guiding practitioners in the design, deployment, and interpretation of deep learning systems, thereby fostering transparency, interpretability, and trust.

Furthermore, as the pace of technological advancement accelerates, the imperative for standardized terminologies and methodologies becomes increasingly salient. An ontological model not only addresses this need but also catalyzes the dissemination of knowledge and best practices, thereby fostering a virtuous cycle of innovation and progress.

In light of these considerations, the development of an ontological model of deep learning neural networks emerges not merely as an academic pursuit, but as a foundational pillar upon which the edifice of contemporary AI is erected. It embodies a convergence of theoretical inquiry and practical application, of disciplinary rigor and interdisciplinary synthesis, of innovation and responsibility. As we embark on this journey of knowledge construction and exploration, we are poised to unravel the mysteries of deep learning neural networks, unlocking new frontiers of understanding and ushering in a future shaped by the transformative power of AI.

Literature Review. In recent years, the field of neural networks has undergone significant advancements, yet many principles articulated by McCulloch remain valid. Despite the diverse range of neuron models available, the fundamental operating principle introduced by McCulloch and Pitts remains unchanged. However, a drawback of McCulloch's model is its reliance on the threshold form of neuron activation function, which limits its flexibility in training and adapting the neural network for different tasks [1].

American neurophysiologist Francis Rosenblatt advanced neural network theory in 1958 by extending McCulloch and Pitts' models. Rosenblatt introduced the "Perceptron," a self-learning neural network with modified connections. Initially featuring a single-layer structure with strict threshold functions and binary or multilevel inputs, the perceptron underwent significant enhancements over time.

In 1982, American biophysicist John Hopfield proposed the Hopfield network, a neural network model incorporating feedback loops between layers to improve generalization properties. This model found extensive application in pattern recognition. Following this, various efficient neural network algorithms and architectures emerged, including backpropagation networks, bidirectional associative memory, and self-organizing maps.

The practical application of neural networks became viable with the rise in computing power, facilitating the widespread use of programs employing neural network principles. Constructing neural networks requires substantial computational resources, given the iterative nature of network training. These networks consist of interconnected nodes with variable weights that acquire knowledge through weight adjustments based on exposed samples. Neurons are linked through network connections, with weights providing instructions in the input signal. Each neuron has its activation level, influencing other neurons and contributing to the overall neural process. To enhance neural processing efficiency, large-scale computers are employed, enabling faster development of computational tasks compared to traditional systems.

In the past three years, there has been a surge in the popularity of deep learning-based methods for process prediction, resulting in significant breakthroughs (Verenich et al., 2019). Notably, pioneering publications on this subject (Evermann et al., 2017; Tax et al., 2017) have amassed over 100 citations, as reported by Google Scholar. Deep learning, characterized by its hierarchical structure of artificial neurons, eliminates the reliance on handcrafted features. This architecture enables artificial neural networks to autonomously learn intricate features, thereby reducing the need for algorithm customization across a diverse range of prediction problems. Moreover, the learned feature representation is not confined by human imagination and can exhibit arbitrary complexity. Additionally, the computational demands and achievable performance of deep learning scale linearly with available data, presenting a contrast to standard process discovery algorithms burdened by quadratic or even exponential runtime requirements (Augusto et al., 2018).

Despite the commonality in the overarching concept of deep learning, the practical implementations for process prediction exhibit fundamental diversity. Variations exist in data preprocessing, the adoption of distinct network architectures, and the pursuit of different objectives such as outcome prediction, time prediction, and control-flow prediction. Furthermore, there is divergence in the sets of log data, evaluation metrics, and baselines used, rendering the results often incomparable. To address this challenge, this paper aims to provide a comprehensive overview of deep learning-based predictive process monitoring approaches, shedding light on the trade-offs among these varied approaches. The contributions of this work are threefold:

A systematic literature review (SLR) is undertaken to construct a comprehensive presentation of existing deep learning-based predictive process monitoring approaches.

The identified literature is systematically classified based on selected criteria to extract the main contributions of each approach.

Conflicting statements and research gaps are highlighted to stimulate further research in this domain.

Materials and methods of research. A deep neural network (DNN) is an artificial neural network with a large number of layers between the input and output layers; each neuron is connected to all neurons in the next layer. One or more layers between the input and output layers are called hidden layers.

Every link between a neuron in a given layer and a neuron in the preceding layer is assigned a weight, denoted as "w," representing the sensitivity of the current neuron's activation to the activation of neurons in the previous layer. Additionally, each neuron in the layer possesses a bias, referred to as "b". If you're familiar with linear regression, the bias term acts as the "c" intercept for $y = mx + c$. If the sum (mx) does not exceed the threshold, but the neuron must be activated, then the predisposition is corrected so that the threshold of this neuron is lowered to activate it [5].

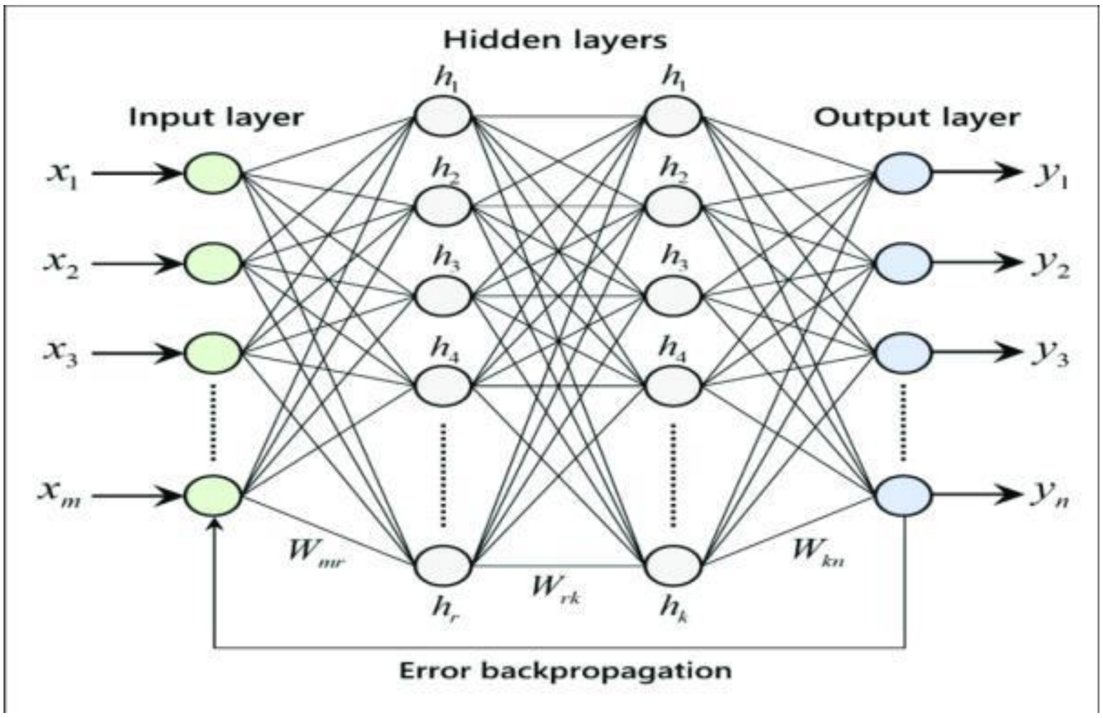


Figure 1. Deep neural network

The whole network seems very complex, doesn't it? But it's not - think of it as a big function y = f(x), where x is the input and y is the output. Then, inside the f(x) function, a number of functions are called, and the output of one function is passed to another. These internal functions are nothing but hidden layers.

The first method uses specially selected data to obtain the desired result. This method is quite laborious, since the data must be collected manually. However, this method is useful for classification and regression. The second method, on the other hand, does not use predefined answers or algorithms. Its purpose is to reveal hidden patterns in the data. This method is commonly used for associative tasks such as clustering or grouping shoppers based on their behavior. "Together with this choose" is one of the types of associative tasks of the Amazon company [6].

Derived from the fundamental idea of a neuron's structure, various mathematical representations of neurons can be constructed. Figure 2 illustrates the prevalent model, wherein the summation function, termed as such, amalgamates all input signals (Xi) originating from the neuron's senders. This summation yields a weighted sum, with the weights (wi) denoting the synaptic strengths. Positive weights are associated with excitatory synapses, whereas negative weights correspond to inhibitory synapses.

The proposed approach relies on employing a knowledge base that corresponds to a unified model designed for representing semantic knowledge. This model utilizes homogeneous semantic networks, specifically those based on the semantic interpretation of basic set theory. This interpretation relies on the membership relation between an element and a set, represented by an arc scale. The languages associated with a unified model for semantic knowledge representation are referred to as sc-languages, with their texts composed of sc-elements. Additionally, sets, structures, ontological representations, and ontological models formed based on sc-languages are termed sc-sets, sc-structures, and sc-models, respectively.

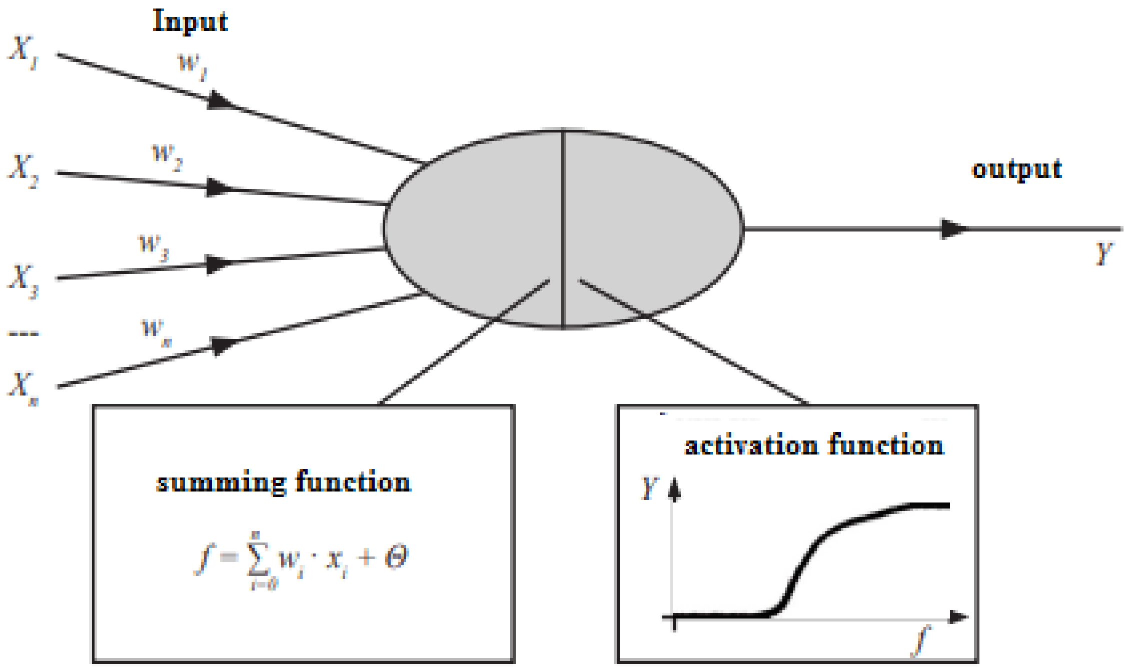


Figure 2. A simple mathematical model of a neuron

Several software systems have been developed for training artificial neural networks. The most widely used are Caffe, Theano, TensorFlow, Torch and CNTK. The Caffe library was the first deep learning system created; it was developed at the Berkeley Center for Vision and Learning and, as of 2014, Caffe contains the largest number of completed and trained models. Theano system was developed at the University of Montreal in Canada. Theano was developed in Python, but Python programs can TensorFlow was created by Google in 2015 and includes a system for efficient processing of tensors and graph flow The Torch library was developed in Lua and provides an easy-to-use high-level environment for creating MATLAB machine learning programs; like Theano, it integrates with the C language for high performance. Torch developers chose Lua over Python because of the ease of integration between C and Lua. In this regard, Microsoft developed CNTK (Cognitive Toolkit) and published its source code in 2016 [7].

Keras: An open source neural network library written in Python, Keras runs on top of open source machine learning libraries such as TensorFlow and Theano The Keras API is divided into three main categories: Models, Layers and Modules They can be divided into the following categories. Unlike the previously described libraries, Keras works with objects without going into the details of creating a mathematical model. For example, to create a convolutional neural network: model = Sequential() model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,28,1)) model.add(Flatten()) model. add(Dense(10, activation='softmax')) model.compile(optimiser='adam', loss='categorical_crossentropy', metrics=['accuracy']) commands: hist = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=1 [8].

Starting a new project. To start a new Python 3 project in Anaconda, you need to select the following menu item – File New Python 3 Notebook

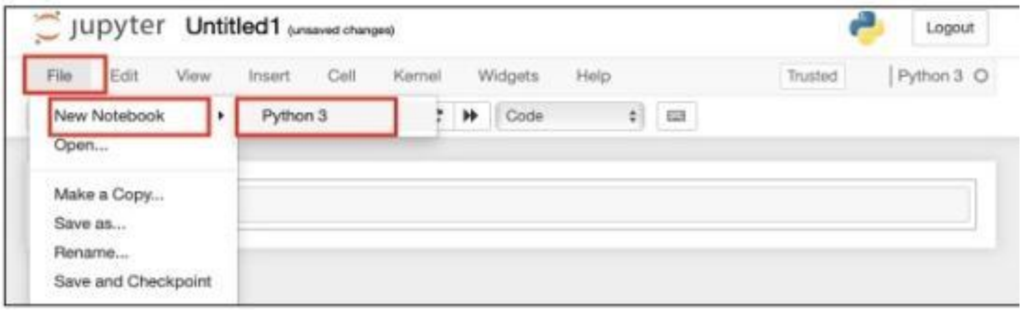


Figure 3. Starting a new project in Python 3

A new empty project should appear on the screen, as shown in Figure 4 below.

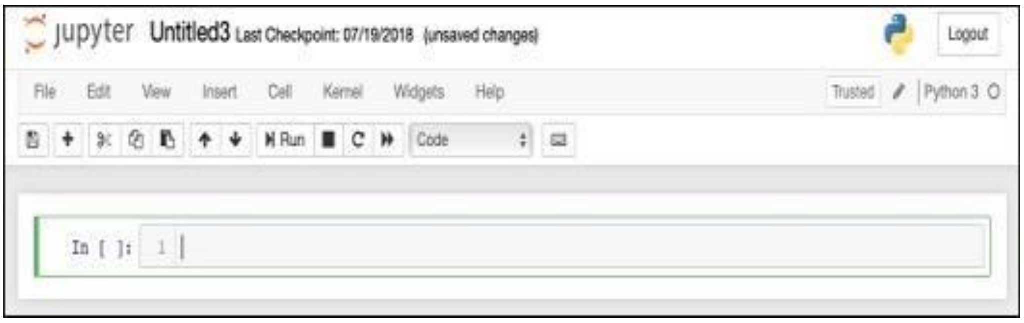


Figure 4. Starting a new project in. Python 3

Rename the project to DeepLearningDigitRecognition by clicking and editing the default name 'UntitledXX'.

Typically, numpy is used for processing arrays and matplotlib for plotting. These libraries were imported into the project using the following import procedure: Since Tensorflow and Keras are constantly updated, if you do not synchronize their respective versions in the project, you will receive many warning errors in the process [9]. This can be distracting from learning, so in this project we will suppress all warnings. This is done with the following lines

```
# disable all warnings
import OS
os.environ['TF_CPP_MIN_LOG_LEVEL']='3'
import alerts
filterwarnings('ignore')
from tensorflow.python. import utilities.
deprecation._PRINT_DEPRECATION_WARNINGS
False
```

Figure 5. Line of code about errors and warnings

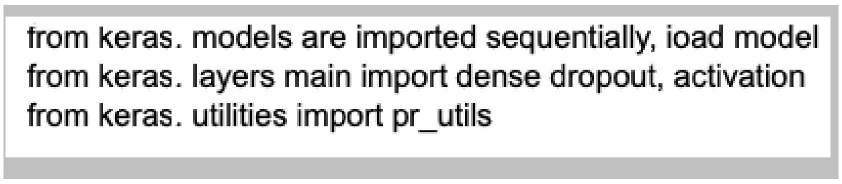
The Keras library is used to import datasets. This uses mnist, a dataset for handwritten digits. The necessary packages are imported using the following instructions - from keras.datasets import mnist. The details presented in the section titled "Materials and Research Methods (table 1).

Table 1. Materials and Research Methods

Research Methods	Description
Deep Neural Network (DNN) Overview	DNN is an artificial neural network with multiple layers (hidden layers) between the input and output layers. Neurons in each layer are connected to all neurons in the next layer. Weights (w) are assigned to each connection, representing sensitivity to activation in the preceding layer. Each neuron has a bias (b), acting like the intercept in linear regression. The bias is adjusted to activate neurons if the sum of inputs does not exceed a threshold.
Simplicity of the Network:	Despite its apparent complexity, the DNN is described as a big function (y = f(x)) with internal functions (hidden layers). Internal functions represent the processing that occurs within the network.
Methods of Data Usage:	Two methods discussed for obtaining desired results: First method involves using specially selected data, useful for classification and regression, but laborious. Second method, not using predefined answers or algorithms, focuses on revealing hidden patterns, often used for associative tasks like clustering.
Neuron Mathematical Representations:	Mathematical representations of neurons are derived from the fundamental idea of a neuron's structure. Figure 2 illustrates a prevalent model with a summation function, weights, and positive/negative associations.
Semantic Knowledge Representation:	The proposed approach involves a knowledge base with a unified model for representing semantic knowledge. It uses homogeneous semantic networks based on the semantic interpretation of basic set theory. sc-languages, sc-elements, sc-sets, sc-structures, and sc-models are key terminologies.
Software Systems for Training Neural Networks:	everal software systems mentioned, including Caffe, Theano, TensorFlow, Torch, and CNTK. Brief descriptions of each system and their origins provided.
Keras Overview:	Keras, an open-source neural network library written in Python, runs on TensorFlow and Theano. Keras API is divided into three categories: Models, Layers, and Modules. Unlike other libraries, Keras focuses on working with objects without delving into the mathematical model details. An example of creating a convolutional neural network using Keras is provided.
Starting a New Project in Anaconda:	Instructions for starting a new Python 3 project in Anaconda are given, involving selecting the menu item "File New Python 3 Notebook."

This section provides a comprehensive overview of the deep neural network, its mathematical representation, methods of data usage, semantic knowledge representation, popular software systems for training, and an introduction to Keras.

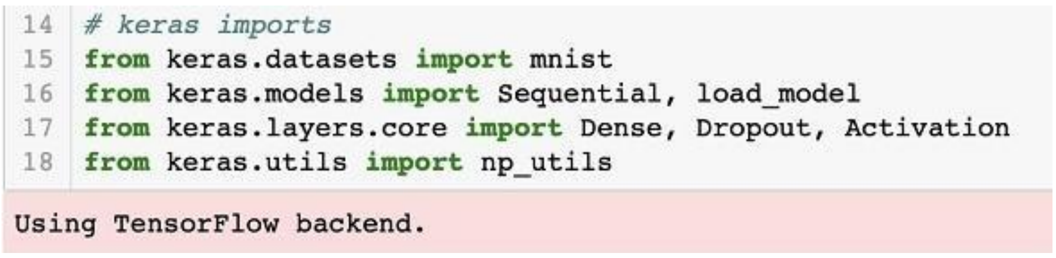
*Results and discussion***.** A deep learning neural network is defined using the Keras package - the Sequential, Dense, Dropout and Activation packages are imported to define the network architecture. load_model package for saving and exporting the model; np_utils is also used for some utilities required by the project [7]. This introduction is done with the help of the following project instructions.



```
from keras. models are imported sequentially, ioad model
from keras. layers main import dense dropout, activation
from keras. utilities import pr_utils
```

Figure 6. Import execution

When you run this code, you will see a console message saying that Keras is using TensorFlow in the back end. A screenshot of this stage can be seen in Figure 7.

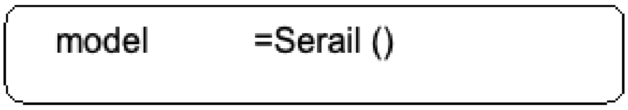


```
# keras imports
from keras.datasets import mnist
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils
```

```
Using TensorFlow backend.
```

Figure 7. Running the code

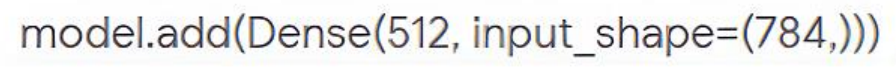
To define a deep learning network architecture, a neural network model will consist of a linear stack of layers. To define such a model, you need to call a sequential function:



```
model        =Serail ()
```

Figure 8. Sequential Function Code Line

The input level is determined, which is the first level in the network, the following step-by-step program instruction is used:



```
model.add(Dense(512, input_shape=(784,)))
```

Figure 9. Line of code defining the input level

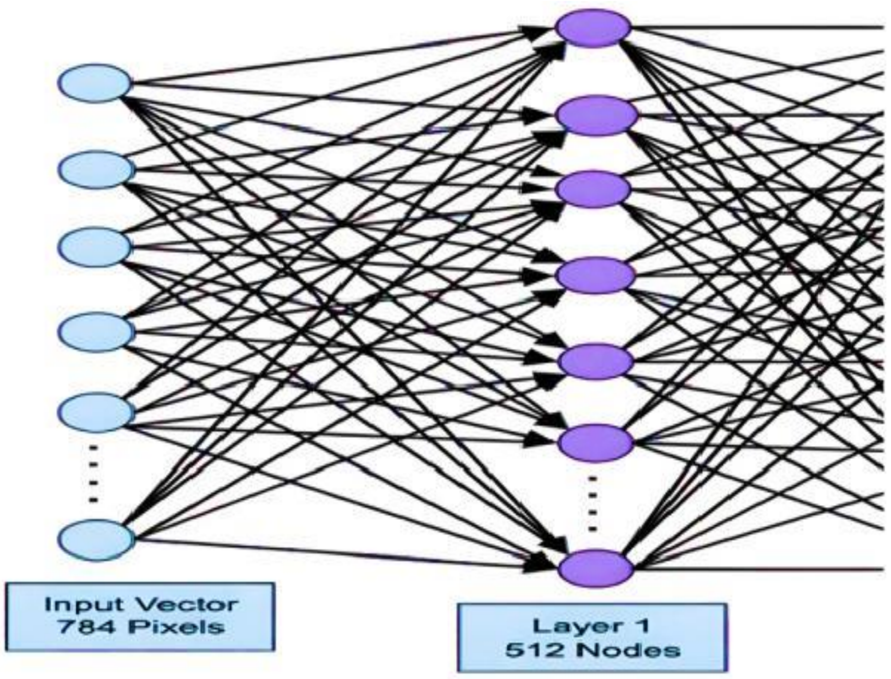
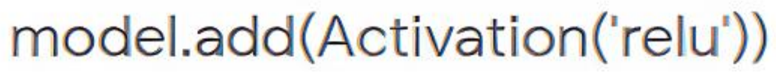


Figure 10. Layer with neurons and input nodes

This creates a 512 node layer (neurons) with 784 input nodes. This can be seen in Figure 10. If you notice that all input nodes are fully connected to layer 1, that is, each input node is connected to all 512 nodes of layer 1.

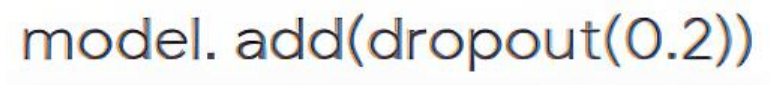
Next, we need to add an activation function to the output of layer 1. We will use ReLU as the activation function. The activation function is added with the following programming command.



```
model.add(Activation('relu'))
```

Figure 11. Activation function

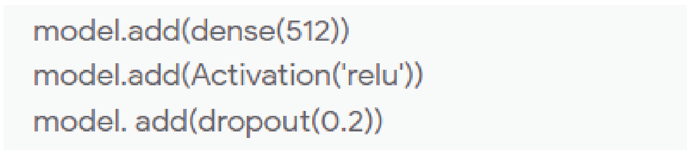
Then we add a 20% dropout using the step-by-step instructions in Figure 12. Dropout is a technique used to prevent the model from overfitting.



```
model. add(dropout(0.2))
```

Figure 12. Dropout method

At this stage, the initial level is completely set. Next, a hidden layer is added. The hidden layer consists of 512 nodes. The input to the hidden layer comes from a previously defined input layer. All nodes are fully connected, as in the previous case. The output of the hidden layer is sent to subsequent layers of the network, namely the final and output layers. Uses the same ReLU activation as the previous layer, downgraded by 20%. The code for adding this layer is shown below.



```
model.add(dense(512))
model.add(Activation('relu'))
model. add(dropout(0.2))
```

Figure 13. Code for adding a layer

The ontological model of the neural network will look like this:

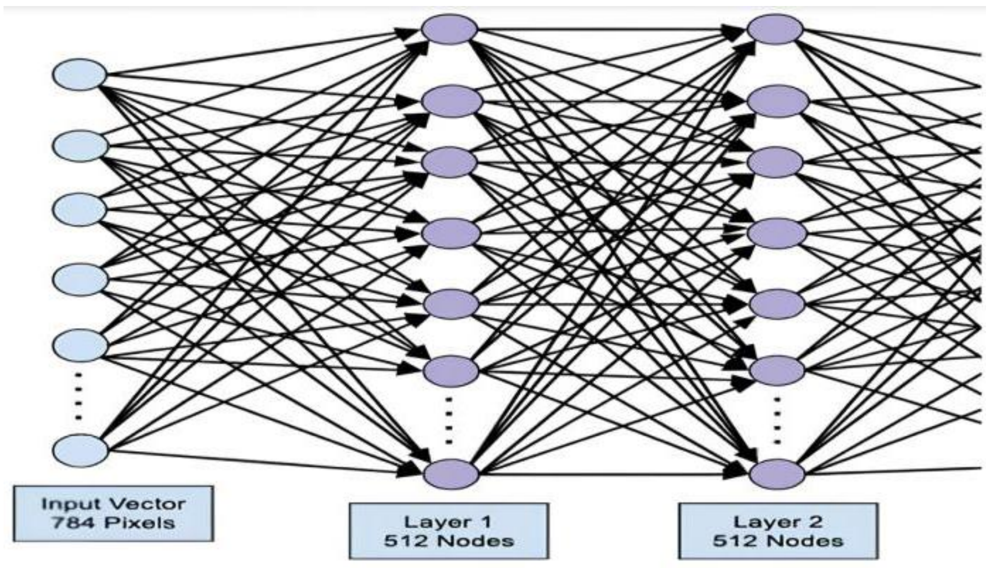


Figure 14. Visualized ontological model of the neural network

Then the last layer is added to this network - the output layer. Note that any number of hidden layers can be added using the same code used here. However, there are some advantages here, as in many cases, but not all, you will get better results.

Conclusion. Since the inception of neural networks, significant transformations have occurred in both their architecture and learning methodologies. Presently, two predominant architectures hold sway: convolutional networks, effectively applied in computer vision tasks, and recurrent networks, actively employed in addressing natural language processing challenges. In the early stages, convolutional networks underwent training through a blend of supervised and unsupervised learning, utilizing automatic encoders and deep belief networks. Modern approaches like residual learning streamline the process by exclusively employing supervised learning, eliminating the need for pre-learning and enhancing the efficiency of learning.

An influential development in convolutional neural networks involves transfer learning, where a network trained on one dataset is adapted to tackle problems in another domain. This entails refining and retraining the network on the target problem's data, reducing training time and broadening the applicability of the trained neural network. Furthermore, the integration of convolutional and recurrent neural networks with reinforcement learning holds promise as a compelling avenue for exploration.

Keras provides a high-level API for building artificial neural networks. In this tutorial, you will learn how to create an artificial neural network trained to find numbers in handwritten text. A layered network has been created for this purpose, and Keras allows you to specify your preferred activation function at each layer. The network was trained on training data using gradient descent. The accuracy of the trained network in predicting unseen data was tested on test data. You will learn how to create metrics for precision and error. Once the network is fully trained, you save the network model for later use. The ontological model of the neural network, visualized in Figure 14, illustrated the structured connectivity between layers. The culmination involved the addition of the last layer - the output layer, providing flexibility for the integration of any number of hidden layers with potential advantages leading to improved results in certain cases. Overall, this research lays the groundwork for the effective application of deep learning neural networks in various applications.

The development of ontological models in the realm of deep learning neural networks represents a paradigm shift in knowledge representation and computational understanding. From foundational concepts to practical implementations, the journey has been marked by continuous innovation. As the field advances, the interplay between ontological models and deep learning algorithms is poised to shape the future of artificial intelligence, making systems more interpretable, transparent, and adaptable to a wide array of applications.

References

1. Kruglov, V.V. Iskusstvennye nejronnye seti. Teorija i praktika / V.V. Kruglov, V.V. Borisov. – M.: Gorjachaja linija – Telekom, 2019. – 382 c.
2. “STIPCompassTaxonomies.pdf.” Accessed: May 27, 2022. [Online]. Available: https://stip.oecd.org/assets/downloads/STIPCompassTaxonomies.pdf
3. E.P. Office, “PATSTAT. Worldwide Patent Statistical Database.” https://www.epo.org/searching-for-patents/business/patstat.html (accessed Mar. 14, 2022).
4. “Database - Eurostat.” https://ec.europa.eu/eurostat/data/database (accessed Mar. 14, 2022).
5. “European innovation scoreboard” European Commission - European Commission. https://ec.europa.eu/info/research-and- innovation/statistics/performance-indicators/european-innovation-scoreboard_en (accessed Mar. 14, 2022).
6. “Scimago Journal & Country Rank” https://www.scimagojr.com/ (accessed Mar. 14, 2022).
7. “Key Enabling Technologies (KETs)” https://knowledge4policy.ec.europa.eu/foresight/-topic/accelerating-technological-change- hyperconnectivity/key-enabling-technologies-kets_en (accessed Mar. 14, 2022).
8. “All Science Journal Classification” Accessed: Mar. 14, 2022. [Online]. Available: https://pg.edu.pl/documents/611754/75313317/asjc
9. “The Enterprise Knowledge Graph Platform | Stardog.” https://www.stardog.com/ (accessed Mar. 14, 2022).
10. “Protégé.” https://protege.stanford.edu/ (accessed Mar. 14, 2022).
11. D. Gasevic, D. Djuric, and V. Devedzic, Model Driven Architecture and Ontology Development. 2006. doi: 10.1007/3-540-32182-9.
12. “Schema.org” https://schema.org/ (accessed Mar. 14, 2022).
13. “DBpedia Spotlight - Shedding light on the web of documents.” https://www.dbpedia-spotlight.org/ (accessed Jan. 08, 2022).
14. Cellfie. Protégé Project, 2022. Accessed: Mar. 14, 2022. [Online]. Available: https://github.com/protegeproject/cellfie-plugin
15. Poveda-Villalón, María, Asunción Gómez-Pérez et al. “OOPS!: An on-line tool for ontology evaluation.” IJSWIS 10.2 (2014): 7-34.
16. LauraCornei, LauraCornei/Onto. 2021. Accessed: Mar. 14, 2022. [Online]. Available: https://github.com/LauraCornei/Onto
17. “IPC V.8 NACE REV.2 Concordance” Accessed: Mar. 14, 2022. [Online]. Available: https://ec.europa.eu/eurostat/ramon/documents/IPC_NACE2_Version2_0_20150630.pdf
18. Augusto A, Conforti R, Dumas M, La Rosa M, Maggi FM, Marrella A, Mecella M, Soo A (2018) Au-to-mated discovery of process models from event logs: review and benchmark. IEEE Trans Knowl Data Eng 31(4):686-705
19. A multi-type semantic interaction and enhancement method for tax question understanding Volume 130, April 2023, 107783 https://doi.org/10.1016/j.engappai.2023.107783
20. BriskilalJ. *et al.* An ensemble model for classifying idioms and literal texts using BERT and RoBERTa Inf. Process. Manage.(2022)