**Implementing Neural Networks using Object-Oriented programming**

A Synopsis Submitted

in Partial Fulfillment of the Requirements

for the Course of

# Minor Project - II

In

Third year – Sixth Semester of

**Bachelor of Technology**

specialization

In

# Artificial Intelligence and Machine Learning

Under

**Dr. Kiran Kumar Ravulakollu**

By

# 500067697 R177218101 Abhiram Varanasi

# 500070807 R177218109 Omair Ansari

# 500067330 R177218043 Hrishabh Punetha

# 500069949 R177218039 Harsh Bhardwaj



DEPARTMENT OF INFORMATICS

SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, BIDHOLI, DEHRADUN, UTTRAKHAND, INDIA

# Jan 2020

Synopsis

1. **Introduction**

Just like the human brain, neural networks are built up of interconnected assembly of simple processing units called nodes which act like neurons in the brain. It is the style and architecture of brain that is incorporated in neural networks [1]. The nodes act as biological neurons and synapses are depicted as single weight so that each input can be multiplied by a weight. The weighted signals are added together to provide node activation. If the node activation is more than the threshold it gives an output 1 or else it gives output 0.

With the availability of huge training datasets and computing power, artificial neural network is widely used today in several applications from classification of skin cancer, image compression and handwriting recognition. Although current neural network models seem to be common enough to apply them to many applications they are still limited to specific problems and impose requirement on the data that is available and interpretability of the solution. Therefore, in some cases, the output or the solution of a neural network model can be restrictive or might be difficult to understand.

Our project’s aim is to overcome this problem. It follows the approach of allagmatic method that programs and executes the model on its own with few limitations while supporting human interpretability. We apply this approach in object-oriented programming to create a metamodel that implements a working dense neural network with variable layer length which makes the models as robust as possible. The input data would guide the selection of a suitable neural network from the metamodel so that possible models would be generated from certain code blocks that are meaningful to humans. Object-oriented programming is selected for the project due to two reasons. First, classes allow the abstract description of structure and operation and objects allow implementing the metastable regime through initialization. Second, they usually provide dynamic and generic types.

1. **Motivation**

Neural Network is an effective learning algorithm for supervised learning and non-linear statistical modeling. However, in some cases, these models might be restrictive on the workable solutions or their solutions might be difficult to interpret. This sort of problem arises because of the “black box” nature of the neural network, which means when one deals with complex neural networks, feedback neural networks while the network can approximate any sort of function, but it won’t provide any insights whatsoever on the structure of function being approximated or about the learnt function from the parameters and therefore are not humanly interpretable. The non-linearity in the activation function and ample number of decisive parameters are one of the reasons for the above problem.

Here we are using concepts of object-oriented programming in determining the relations between the various functionalities of a neural network, creating classes, objects, class diagrams to best fits the relations on OOP’s and thus implementing and the obtained model on various datasets.

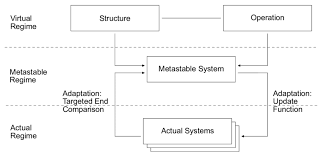
The goal is to implement a working dense neural network with variable layer length using the object-oriented programming paradigm while supporting human interpretability and making it more robust at the core.

1. **Related work**

We plan to design a dense neural network with variable number of layers for a few problem cases using OOP concepts. Our network shows the emphasis of weights and various functions and factors involved in the functioning of the network and highlight the correlation between these functions, factors and weights using simple OOP concepts.

Many models provide solutions that are accurate but also difficult to interpret, according to this [2] paper to overcome this particular problem allagmatic method which is autonomous not only keeps limitations and margin for error to minimal but also making the solution much more interpretable. Metamodel and its building blocks are entity and update function that describes a computer model. And by automatically combing these blocks the interpretability might be increased. In this [2] object-oriented programming is proposed to implement the entities and their milieus as dynamic and generic arrays and the update function as a method.

[3] The allagmatic method consisting of a system metamodel that is abstractly described in the virtual regime, concretized with parameters in the metastable system, and run or executed in the actual regime. Adaptation as introduced in this study is occurring between the metastable and actual regime.



*fig 1: Gilbert Simondon’s philosophical concepts applied to meta-modelling of complex systems.[2]*

1. **Methodology**

The entire project has been divided into four parts, shown in fig 2. The middle part containing Helper functions and Class schema are used to make the neural network. Here usage of object-oriented programing helps designing a better and optimal classes for neural network integration. The other modules help in represent the output more interpretable. These uses various common and standard statistical techniques for analyzing and representation of the provided input. In case meta model module the input would clean train data. For Representation module, it would be final network after training.

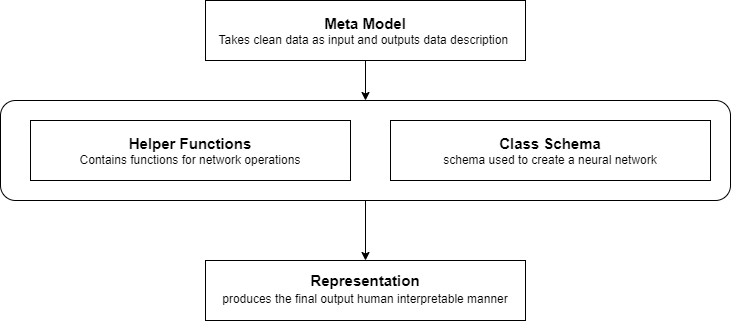
**Note: -** Some Specifications and Assumptions used in this project need to be acknowledged.

**Specifications:**

Our model is for creation of dense neural network. Works with specific hyperparameters.

**Assumptions:**

The data provided to the model is clean and ready for training.

*fig 2: The General Methodology of the Project*

1. **Plan of work**

|  |  |
| --- | --- |
| **Month** | **Work Description** |
| January | **Ideation:** Brainstorming to find a problem statement, that can be done in the stipulated time.  **Research:** Finding resources like research papers, blogs, patents related to the problem statement that we have chosen. |
| February | **Solution Building:**  Designing solution to the problem statement we have chosen and with the resources, we have gathered.  **Methodology:** Designing proper method of the solution and dividing the problem into modules.  **Algorithm:** Designing algorithm/flow chart for each module. |
| March | **Coding:**  Module wise coding, testing, and debugging errors.  **Integration:** Integrating individual modules into a single file and testing it and rectifying errors. |
| April | **Testing:**  Testing the algorithm, finding time and space complexities.  **Documentation:** Explaining the code and different modules. Preparing a document that summarizes the work, with figures and test results and interpretation of test results. |
| May | **Presentation:** Preparing a presentation that encapsulate the whole project, which consists of problem statement, need, solution and summary of documentation. |

**NOTE:**

1. Include diagrams/images and tables as per the requirement of the project. Images must be high resolution so that it must not look blurry.
2. Adhere with the format without any modification.
3. Follow the synopsis submission deadline.

**References**

[1] Braspenning, P., 1995. *Artificial neural networks*. Springer.

[2] Patrik Christen and Olivier Del Fabbro, 2020. *Automatic Programming of Cellular Automata and Artificial Neural Networks Guided by Philosophy*. Springer, pages 131–146.

[3] Christen, Patrik & Fabbro, Olivier. (2020). *Adaptation in a System Metamodel for Evolutionary Computation*. [online] Available at: <https://www.researchgate.net/publication/344038715_Adaptation_in_a_System_Metamodel_for_Evolutionary_Computation>