

GUI BASED PYTHON CODE GENERATOR FOR CNN

Submitted in partial fulfilment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering
by

ABHISHEK (38110007)
MAYANK KUMAR SINGH (38110312)



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE
AND TECHNOLOGY (DEEMED TO
BE UNIVERSITY)**

**Accredited with Grade "A" by
NAAC JEPPIAAR NAGAR,
RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MARCH 2020



**SATHYABAMA INSTITUTE
OF SCIENCE AND
TECHNOLOGY (DEEMED TO
BE UNIVERSITY)**

Accredited with “A” grade by NAAC Jeppiaar
Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Abhishek (38110007)** and **Mayank Kumar Singh (38110312)** who carried out project entitled “**GUI BASED PYTHON CODE GENERATOR FOR CNN**” under my supervision from November 2019 to April 2020.

Internal Guide

Mr. R. KAMALESH M.E.

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Abhishek (38110007)** hereby declare that the Project report entitled “**GUI BASED PYTHON CODE GENERATOR FOR CNN**” done by me under the guidance of **Mr. Kamallesh. M.E.** is submitted in partial fulfilment of the requirements for the award of **Bachelor of Engineering degree** in the **Department of Computer Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

DECLARATION

I **Mayank Kumar Singh(38110312)** hereby declare that the Project report entitled **“GUI BASED PYTHON CODE GENERATOR FOR CNN”** done by us under the guidance of **Mr. Kamallesh M.E.** is submitted in partial fulfilment of the requirements for the award of **Bachelor of Engineering degree** in the **Department of Computer Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Boards of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sasikala M.E., Ph.D., Dean of the Department, School of Computing**, for providing us necessary support and dataset at the right time during the progressive reviews.

We convey our thanks to **Dr. S. Vigneshwari M.E. , Ph.D. , Head of the Department, Department of Computer Science & Engineering**, for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Mr Kamalesh M.E., (Ph.D.)**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science & Engineering who** were helpful in many ways for the completion of the project.

ABSTRACT

The data mining is a process that is basically used to mine the data and give the result that are hidden to the internal database. The data mining is done in very formal that are basically used in medical field, engineering field and quite also in technical field. The data mining basically uses the machine learning algorithm which are predictable in nature. The heart disease prediction is basically a process which took some of the information from the user and then mine the data to predict the answer i.e, it has heart disease or not. Following are some data mining technique that are used for the prediction. These are Random Forest Decision Tree & Naive Bayes etc. from the algorithm procedure it is formed the Random Forest has the best accuracy and precision with 81% when compared to the algorithm for heart disease prediction.

Title	Page No.
TITLEPAGE	
UNDERTAKING	
CERTIFICATE OF ORIGINAL WORK	
CERTIFICATE FROM EVALUATION COMMITTEE	
ABSTRACT	I
ACKNOWLEDGEMENT	II
TABLE OF CONTENTS	III
LIST OF FIGURES	V
LIST OF TABLES	V
CHAPTER I	INTRODUCTION
1.1 Overview	1
1.2 Objectives	1
1.3 Product Perspective	1
1.4 Product Functions	2
1.5 User Classes & Characteristics	2
1.6 Operating Environment	3
1.7 Existing Technologies	3
CHAPTER II	REQUIREMENT ANALYSIS
2.1 Functional Requirements	5
2.2 Non-Functional Requirements	7
2.3 Hardware/Software/Communication Requirements	8
2.4 Software Quality Attributes	9
CHAPTER III	SYSTEM DESIGN
3.1 System Design	10

3.2	Architectural Design	22
3.3	UI/UX Design	23

CHAPTER IV CODING AND TESTING

4.1	Sample Coding	24
4.2	Test Strategy and Approaches	31

CHAPTER V RESULTS AND DISCUSSION

5.1	Screenshots	32
-----	-------------	----

CHAPTER VI CONCLUSION AND FUTURESCOPE

6.1	Conclusion	36
6.2	Future Scope	36

REFERENCES	37
-------------------	-----------

LIST OF FIGURES

Figure no.	Caption	Page No.
Figure 1	Build Section Design	11
Figure 2	Activation Functions	13
Figure 3	Linear Function Graph	14
Figure 4	Sigmoid/Logistic Graph	14
Figure 5	Hyperbolic Tangent Graph	15
Figure 6	Rectified Linear Unit Graph	15
Figure 7	Learn Section Design	16
Figure 8	Train Section Design	20
Figure 9	Epochs Graphs	21
Figure 10	Architecture Design	22
Figure 11	UI/UX Design and Layout	23
Figures 12	Testing : JEST	31
Figure 13	Test Results	31
Figure 14	PyCoGen Screenshots 1-4	32
Figure 15	Demonstration Screenshots 5-7	34

LIST OF TABLES

Table No.	DESCRIPTION	PAGE. NO
Table 1	Functional Requirement elements	5
Table 2	Software Interfaces	8

CHAPTER-I

INTRODUCTION

1.1 Overview

Python code Generator (PYCOGEN) serves the purpose of Low code Generator tool for CNN Architecture using Graphical User Interface(GUI). This tool provides the platform that aids in the better interaction of the User with an interactive GUI that visualize complex Convolutional Neural Network (CNN) Architecture and that makes it possible to generate CNN model code in Python, which can further be implemented in Users machine learning and deep learning projects

1.2 Objectives

The objective of this software is:

- ❖ To design and develop an in-browser Code Generator low code tool for generating ~~python~~ ~~code~~

Which is:-

- Interactive
 - Easy to use
 - Modular
-
- ❖ To develop a GUI based tool to visualize and generate CNN model code in python language.

1.3 Product Perspective

CNN is a broad field and it requires heavy parameter optimization and experimentation before arriving at a particular result. That being said, if you want to take a more automated approach to CNN architecture design, look up "neural architecture search" and "neuro-evolution". These approaches tend to be pretty heavyweight (i.e., you're going to need a cluster) but the idea is to automate the design process to find an "optimal" network. Neural architecture search strategies tend to use machine learning approaches (e.g., recurrent neural networks to generate the structure, or reinforcement learning) while neuro-evolution strategies rely on evolutionary algorithms.

Our Tool allows for the development of Architectures straight on the Web Browser without writing any Code which can be experimented with and used for our Deep Learning Project.

1.4 Product Functions

This Product plays a major role in generating the python Code of a CNN architecture model as required by the user. The product helps developers visualize CNN model architecture structures as well as the processes of internal feature abstractions, and final inference generation code. This Product is interactive for changing and making alterations according to the user requirement.

For working on a machine & deep learning projects where user (most likely programmers and developers) are challenged to face the use of Convolutional neural network model in their product to make their product competitive and more appealing, in this conduct our product gives an advantage to the user by making a tedious task hassle free and more customized according to needs and requirement.

The key functions of the product is to generate the CNN Architecture base model by giving user an interactive user interface to deal with their approach for the deep learning or machine learning projects and researches the key noted feature are as follows:

- Have a Build Section where the Layers can be added and Removed. In addition, we have a Node Count where we can increase the Nodes. We can also set up Activation Functions and Weight Initializers along with it.
- Have a Learn Section where the Optimizer Function can be set along with the Loss Function. We also have a Learning Rate Specifier and a Learning Rate Decay Parameter.
- Have a Train Section where we can specify the number of Epochs till which we can train the Model and the Batch Size.
- Once we specify all these features, we will get a generated code which can be used in our Neural Network Architecture for Data Modeling. We can also visualize the Neural Network automatically on the Browser UI itself for the better purpose.

1.5 User Classes and Characteristics

- Typical Users, such as students/scholars, who want to use this tool for analyzing and generating architecture model code for convolution neural networks(CNN).

- Advanced/Professional Users, such as data scientists and data engineers or researchers, who want to use this software for more demanding coding and analysis implementations.
- Programmers who are interested in working on the project by further developing it or fix existing bugs.

1.6 Operating Environment

The working Environment for the Code Generator requires the least configuration of all as it is a web based In-Browser system. Users need to have a python working environment to implement the generated code or e.g. Jupyter Notebook, Google colab and other IDE's. While bandwidth and hardware upgrades may most likely be required, the current architecture can largely be maintained.

The website is expected to be hosted on a build server that is in-house and resides on hardware that will be maintained by the Host. This can be executed in the form of a virtual machine to allow growth for website traffic. Having the web server run on a virtual machine will allow for extremely easy migration of the web server in the event that the owner wishes to outsource web hosting to an external company. A build server is also expected to reside on the virtual machine for the purposes of storing and retrieving system data.

Due to the web-based nature and of the system, it will be largely compatible with any device that is capable of internet access and has a web browser JavaScript enabled with JSX support except Internet Explorer. The website will be coded with industry standard practices to keep compatibility issues to a minimum.

The most popular web browsers except Internet Explorer in the world are recommended as they privilege special care and ensure optimum compatibility. As such, all latest browsers are recommended for use.

For developers, programmers, testers and host the operating requirement needs to have a running system with Node.js enabled if deployed on local machine and on a build server it is to be deployed on internet and domain of selection which suits the requirement and reach of the targeted audiences.

1.7 Existing Technology

1) - Pix-to-code: Generating Code from a Graphical User Interface Screenshot

Transforming a graphical user interface screenshot created by a designer into computer code is a typical task conducted by a developer in order to build customized software, websites, and mobile

applications. In this paper, we show that deep learning methods can be leveraged to train a model end-to-end to automatically generate code from a single input image with over 77% of accuracy for three different platforms (i.e. iOS, Android and web-based technologies).

2) -Cog

Cog is a simple code generation tool written in Python. We use it or its results every day in the production of Kubi.

Kubi is a collaboration system embodied in a handful of different products. We have a schema that describes the representation of customers' collaboration data: discussion topics, documents, calendar events, and so on. This data has to be handled in many ways: stored in a number of different data stores, shipped over the wire in an XML representation, manipulated in memory using traditional C++ objects, presented for debugging, and reasoned about to assess data validity, to name a few.

3) -Automatic GUI Creation:

4) Generating Java source code from formal descriptions

5) <https://www.percederberg.net/text/master/page04.html>

6) **-Blockly** is a client-side library for the programming language JavaScript for creating block-based visual programming languages (VPLs) and editors. A project of Google, it is free and open-source software released under the Apache License 2.0.^[2] It typically runs in a web browser, and visually resembles the language Scratch. It is also being implemented for the mobile operating systems Android and iOS, though not all of its browser-based features will be available on those platforms.

Blockly uses visual blocks that link together to make writing code easier, and can generate code in JavaScript, Lua, Dart, Python, or PHP. It can also be customized to generate code in any textual programming language.

CHAPTER-II

REQUIREMENT ANALYSIS

2.1 Functional Requirements

The Functional Requirement for PyCoGen elements are as follows:-

Note:-Please refer to the element Dictionary table for all the elements that are to be displayed on the screens

Sr.No	Name	Description	Control Type	Default Value	Validation
1	Add Layer	To add the layer of ConvNet	Radio	1	when this option is selected (a) - if no layer exists add the layer (b) - add the layer to the previous layer of the ConvNet
2	Remove Layer	To Remove the Layer of ConvNet	Radio	0	When the option is selected the layer are to be removed from the ConvNet until there is 0 layers
3	Node Count	To enter the node count on each layer	Slider	5	To set the number of nodes in each layers for the Convnet
4	Activation Function	To activate the nodes and layers of the ConvoNet	Dropdown	relu	when this option is selected the user can select from the available options and all parameter instantiates from a configuration library
5	Weight Initializers	To initialize the Weights of each neuron	Dropdown	glorot uniform	when this option is selected the user can select from the available options and all parameter instantiates from a configuration library
6	Optimizers	To optimize and to deal with optimization problems by minimizing the function..	Dropdown	SGD	when this option is selected the user can select from the available options and all parameter instantiates from a configuration library
7	Learning Rate	The amount that the weights are updated during training is defined as the step size or the "learning rate."	Dropdown	Mean Squared Error	when this option is selected the user can select from the available options and all parameter instantiates from a configuration library
8	Learning Rate of the Network	The Learning rate of the entire Convnet model	Slider	0.001	To set the Learning rate of Convnet Model varying from 0-1
9	Learning Rate Decay	It can be defined as the way in which the learning rate changes over time and is also known as learning rate schedule	Checkbox	N/A	To introduce learning rate decay in a ConvNet by 0.01
10	Epochs	Epoch is when an entire dataset is passed forward and backward through the neural network only once	Slider	100	To set the no of epochs for training the ConvNet varying from 0-500 epochs according to the user requirements
11	Batch Size	The batch size impacts how quickly a model learns and the stability of the learning process.	Slider	32	To set the no of batch size for enhancing the stability of the models learning process by
12	Generated Code	To show the Generated code of ConvNet on a Display Console	N/A	Basic Syntax	To show the Generated code on a panel from the desired ConvNet
13	Model Architecture	To show the ConvNet Model preview of the desired Input	N/A	N/A	To provide a visual and pictorial representation of the desired ConvNet for which the Code is generated

Table 1

❖ **Add Layer and Remove Layer:-** The Add/Remove layer features in the build sections requires the user interaction to provide an input to the system for the no of layer with each layers node count defined as desired the user

Node count:- the default values of the node count is defined as 5, but it can be altered by the user through a slider according to the desired values

The layer and node count sections input are served to the visualizers and algorithms where the layers and node is being previewed

❖ **Activation Functions:-** The user and the client are required to select the activation function from the available options for activation of the nodes in a layer ,although the default values

are pre-defined. the Activation functions are instantiating from a configuration library of keras as an high level API and tensorflow.js the activation functions alteration done by the user gets updated in Generated Code.

- ❖ **Weight Initializers:-** The user and clients are required to add weight on each layers the default values are pre-defined, the user can choose and alter from the available options. All the weights and biases are instantiating from a configuration library of keras as an high level API and tensorflow.js the weights alteration done by the user gets updated in Generated Code.
- ❖ **Optimizers:-** The user and clients are required to choose from available options in optimizers, the default values are pre-defined. All hyper parameters are instantiating from a configuration library of keras as an high level API and tensorflow.js the weights alteration done by the user gets updated in Generated Code.
- ❖ **Loss Functions:-** The user and clients are required to choose from available options in Loss functions , the default values are pre-defined,. All hyper parameters are instantiating from a configuration library of keras as an high level API and tensorflow.js the weights alteration done by the user gets updated in Generated Code.
- All hyper parameters are instantiating from a configuration library of keras and Tensorflow,the retrieving and making use of the algorithms are solely based on their configuration resources
- The most popular web browsers except Internet Explorer in the world are recommended as they privilege special care and ensure optimum compatibility.
- Due to the web-based nature and of the system, it will be largely compatible with any device that is capable of internet access and has a web browser JavaScript enabled with JSX support. The website will be coded with industry standard practices to keep compatibility issues to a minimum. Since the product is developed in ReactJS which has the JSX syntax format, the Browser Internet Explorer is incapable of serving because it does not support JSX.

2.2 Non-functional Requirements

Performance Requirements

The system is interactive and the delays involved are less. So in every action-response of the system, there shall be no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, In case of evaluation there are no delays and the operation is performed in less page load time measured 0.9 sec, computing, posting > 95% of the files. Also when connecting to the build server the delay is based on the distance of the 2 systems and the configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for sake of good communication .Web application on any browser except internet explorer (because it doesn't support JSX code(*JSX is an XML/HTML-like syntax used by React that extends ECMA Script so that XML/HTML-like text can co-exist with JavaScript/React code.*)). The system is 100% Responsive to any screen size devices for client side. The Performance and accuracy live speed response of product is expected if deployed on AWS Amplify (recommended) at mobile devices rate 79-80/100 and at the desktop devices 97-98/100.

Safety Requirements

If there is a Build server failure or malfunctioning, website crash damage to a wide portion of the System files due to catastrophic failures such as updates and version releases, the Restore Points of the previous stable version /updates will be restored and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failures.

This product is safe for all users mentioned above such as data scientist researchers and scholars, etc. This product does not store any personal information that is retrieved from the user and therefore there isn't a possibility of exposure of personal information. The data is not stored even at browser in the form of cookies, etc.

Security Requirements

Since the product specified is a web based therefore proper precautions and safety measures must be taken by the user to thwart and prevent cyber-attacks like phishing and Denial-of-service(DoS) and distributed denial-of-service (DDoS) attacks,etc.

A secure Internet connection from a reliable internet service provider must be procured for the smooth and safe running and usage of the web based tool.

The network security services will also be running in the backend which will trace and keep track of system integrity. The system includes authentication gateway for the administrator/ host(only for build several iterations) of the web site provided by hosting vendor , although it would probably

be free hosted website yet there are no chances that any random user can disturb the system files. server side security and website security provided by react and vendor assure that product would be secured and confidential

2.3 Hardware/Software/Communication Requirements

Hardware Interfaces

- Client machine with any standard OperatingSystem
- A browser which supports HTML &JavaScript (ES6) andJSX
- Internetconnectivity

Software Interfaces

The product software interfaces uses ReactJS, NodeJS, Tensorflow, and Keras.

The Software Interfaces of the product development software that are taken into consideration are as follows :-

Table 2

Software used	Description
Operating system(OS)	Any Widely used operating system are recommended but for demonstration its previewed in Windows 10 for its best support and user-friendliness.
ReactJS	React is an open-source, front end, JavaScript library for building user interfaces or UI components.
Tensorflow	TensorFlow.js works in the browser and Node.js, and in both platforms there are many different available configurations.
NodeJS	To run on the browser-side the NodeJS is the suitable library to run Javascript
Keras	The Layers API of TensorFlow.js is modelled after Keras and we strive to make the Layers API as similar to Keras as reasonable given the differences between JS & Python.

Communications Interfaces

This Product supports all types of web browsers except internet explorer. We are using React JS for designing the Graphical User Interface and its interiors etc. for other communication tensorflow.js will be interacting with Keras as high level API and others for computational parts where as NodeJS will be interacting at the run time with the system to generate the desired code of the CNN architecture model. Web browsers and servers communicate using TCP/IP. Hypertext Transfer Protocol is the standard application protocol on top of TCP/IP supporting web browser requests and server responses. Like most internet traffic, web browser and server connections normally run through a series of intermediate network routers.

Software Quality Attributes

Availability

If the internet service gets disrupted while sending information to the server, the information can be send again for verification.

Security

The main security concern is for users as this products system security integrity is kept secure at server side. Hence, security is provided from unwanted use of breach at server side and system integrity software.

Usability

As the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

Portability

This system will be portable to different systems environment and easily accessible.

Adaptability

This system will not be adaptable to different data formats and can be used in only in the course of specified area.

Maintenance

The system is easy to Maintain by the Host however for updates and new features the system is easy to maintain

CHAPTER-III

SYSTEM DESIGN

3.1 SystemDesign

Criticality Scaling criteria :Low (1) – Medium (2) – High (3)

1. **Low:** Items that can be eliminated should the need arise, without adversely affecting the product. These functions/Design are not urgent but are important to finalproduct.
2. **Medium:** Items that are desired by the customer and/or users of the system, but that may be postponed until a future release. These items are not urgent but are important parts of the final product.
3. **High:** Items that are missing critical and without which the system cannot function in a manner that is satisfactory to the customer. These items are urgently needed and important to the success of the final product.

Priority Scaling criteria:Low (2) – High (1)

1. **Low** –the activity which is not highly recommended or will be required by any user while using theproduct
2. **High**– the activity which is necessarily required to be fulfilled, without which the purpose of the product won't be served.

System Design 1

Have a Build Section where the Layers can be added and Removed. In addition, we have a Node Count where we can increase the Nodes. We can also set up Activation Functions and Weight Initializers along with it.

Node Count and Layers

Artificial neural networks have two main hyperparameters that control the architecture or topology of the network: the number of layers and the number of nodes in each hidden layer.

You must specify values for these parameters when configuring your network.

A node, also called a neuron or Perceptron, is a computational unit that has one or more weighted input connections, a transfer function that combines the inputs in some way, and an output connection.

Nodes are then organized into layers to comprise a network.

A single-layer network can be extended to a multiple-layer network, referred to as a Multilayer Perceptron. A Multilayer Perceptron, or MLP for short, is an artificial neural network with more than a single layer.

It has an input layer that connects to the input variables, one or more hidden layers, and an output layer that produces the output variables.

We can summarize the types of layers in an MLP as follows:

Input Layer: Input variables, sometimes called the visible layer.

Hidden Layers: Layers of nodes between the input and output layers. There may be one or more of these layers.

Output Layer: A layer of nodes that produce the output variables..

Finally, there are terms used to describe the shape and capability of a neural network; for example:

Size: The number of nodes in the model.

Width: The number of nodes in a specific layer.

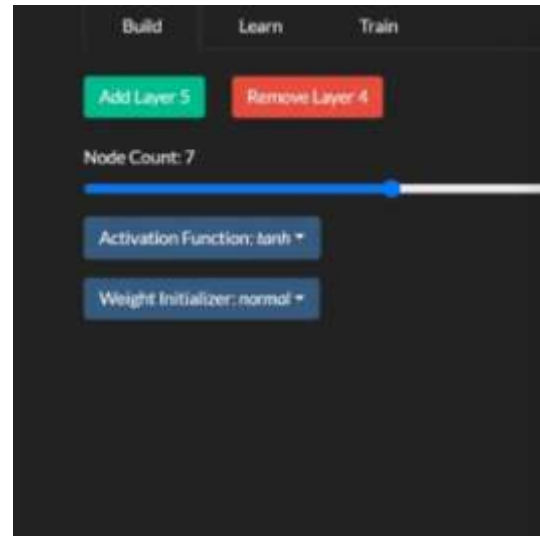


Figure 1: Build Section

Depth: The number of layers in a neural network.

Capacity: The type or structure of functions that can be learned by a network configuration. Sometimes called “*representational capacity*”.

Architecture: The specific arrangement of the layers and nodes in the network.

Weight Initializers

The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing due to use of a forward pass through a deep neural network. If either occurs, loss gradients will either be too large or too small to flow backwards beneficially, and the network will take longer to converge, if it is even able to do so at all.

All Weight Initializers instantiate from a configuration dictionary. The

weight initializers are as follows:-

- **ZEROS**

Initializer that generates tensors initialized to 0

- **ONES**

Initializer that generates tensors initialized to 1.

- **GLOROT UNIFORM**

The Glorot uniform initializer, also called Xavier uniform initializer.

Draws samples from a uniform distribution within [-limit, limit], where $limit = \sqrt{6 / (fan_in + fan_out)}$ (fan_in is the number of input units in the weight tensor and fan_out is the number of output units).

- **HE UNIFORM**

He uniform variance scaling initializer.

Draws samples from a uniform distribution within [-limit, limit], where $limit = \sqrt{6 /$

fan_in) (*fan_in* is the number of input units in the weight tensor).

- **NORMAL**

Initializer that generates tensors with a normal distribution..

Activation Function

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

We know, neural network has neurons that work in correspondence of weight, bias and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

It doesn't matter how many hidden layer we attach in neural net, all layers will behave same way because the composition of two linear function is a linear function itself. Neuron cannot learn with just a linear function attached to it. A non-linear activation function will let it learn as per the difference with respect to error. **Hence we need activation function.**

- Activation Functions are mathematical equations that determine the output of a neural network attached to each neuron in the network, and determine whether it should be activated ("fired") or not, based on whether each neuron's input is ~~relevant~~ relevant for the model's prediction.



- They must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample.

It is used to determine the output of neural network like yes or no. It maps the resulting values inbetween 0 to 1 or -1 to 1 etc. (depending upon the function).

FOLLOWING ARE THE ACTIVATION FUNCTION

● **Linear Function**

- Linear Function and Derivative takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input.
- In one sense, a linear function is better than a step function because it allows multiple outputs, not just yes and no.

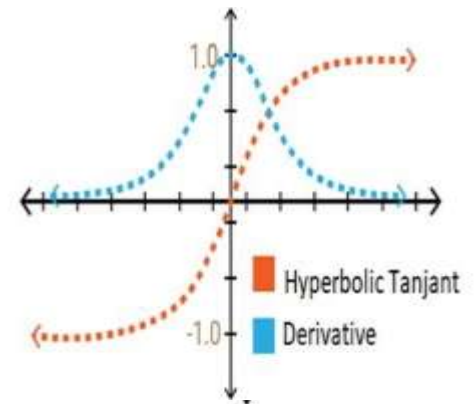


Figure4

● **Sigmoid/logistic**

- The function takes any real value as input and outputs values in the range 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0.
- especially used for models where we have to **predict the probability** as an output
- It is most frequently used activation function
- The function is **differentiable, monotonic, Smooth gradient, Output values bound, Clear predictions**

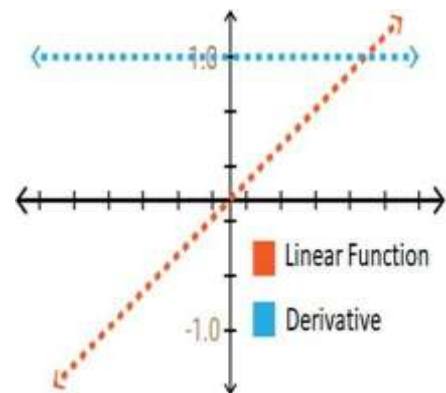


Figure4

- **Hyperbolic Tangent(Tanh)**

- Tanh function is defined as $(-1, +1)$, **it is zero-centered**.
- The advantage over the sigmoid function is that **its derivative is more steep**, which means it **can get more value**. This means that it will be **more efficient because** it has a **wider range for faster learning and grading**.

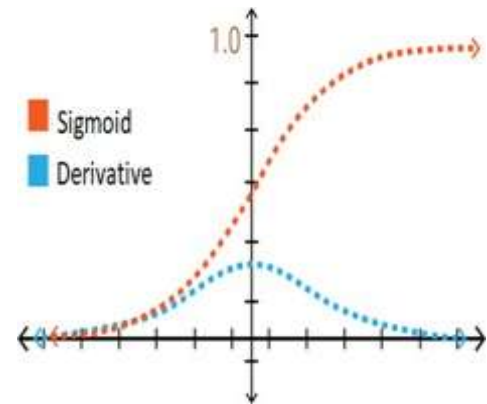


Figure 5

- **ReLU (Rectified Linear Unit).**

- ReLU activation function, is perhaps the most common function used for hidden layers.
- simple to implement and effective at overcoming the limitations of other activation functions, such as Sigmoid and Tan h.
- Frequently applied in convolutional neural networks or deep learning.
- **Computationally efficient—ReLU is valued at $[0, +\infty]$**

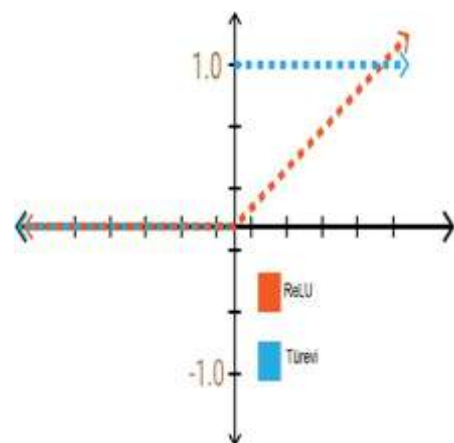


Figure 6

- **Softmax**

- Softmax is frequently used as an activation function in the output layer of neural network
- used when the model internally needs to choose or weight multiple different inputs at a bottle neck or concatenation layer.

Critical Scale: 3

Priority Scale: 2

System Design 2

Have a Learn Section where the Optimizer Function can be set along with the Loss Function. We also have a Learning Rate Specifier and a Learning Rate Decay Parameter.

Optimizer

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function.

Optimization algorithms are responsible for reducing the losses and to provide the most accurate results possible.

The weight is initialized using some initialization strategies and is updated with each epoch according to the update equation.

$$W_{new} = W_{old} - lr * (\nabla_W L)_{W_{old}}$$

The above equation is the update equation using which weights are updated to reach the most accurate result. The best result can be achieved using some optimization strategies or algorithms called optimizers.

The Optimizers included in the project are as follows:-

- **SGD(STOCHASTIC GRADIENT DESCENT)**

- It is a variant of Gradient Descent.
- It tries to update the model's parameters more frequently.
- In this, the model parameters are altered after computation of loss on each training example.
- It update the model parameters one by one. If the model has 10K dataset SGD will update the model parameters 10k

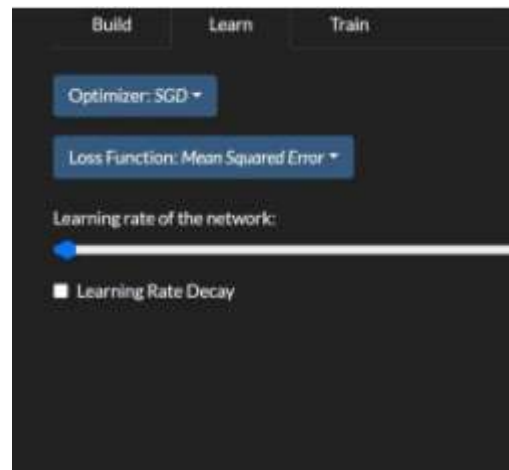


Figure 7: Learn Section

- **RMS-Prop (Root Mean Square Propagation)**

- RMS-Prop is a special version of Adagrad in which the learning rate is an exponential average of the gradients instead of the cumulative sum of squared gradients.
- In RMS-Prop learning rate gets adjusted automatically and it chooses a different learning rate for each parameter

- **AdaDelta**

- Adadelta is an extension of Adagrad and it also tries to reduce Adagrad's aggressive, monotonically reducing the learning rate and remove decaying learning rate problem.
- In Adadelta we do not need to set the default learning rate as we take the ratio of the running average of the previous time steps to the current gradient.

- **Adam (Adaptive Moment Estimation)**

- Adam optimizer is one of the most popular and famous gradient descent optimization algorithms.
- It is a method that computes adaptive learning rates for each parameter.
- It stores both the decaying average of the past gradients, similar to momentum and also the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods.

-

Loss Function

Neural networks are trained using stochastic gradient descent and require that you choose a loss function when designing and configuring your model.

In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the objective function.

We may seek to maximize or minimize the objective function, meaning that we are searching for a candidate solution that has the highest or lowest score respectively.

Typically, with neural networks, we seek to minimize the error. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply “loss.”

The cost or loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model.

Loss functions can be classified into two major categories depending upon the type of learning task we are dealing with — **Regression losses** and **Classification losses**.

Regression loss Functions:-

A Regression predictive modelling problem involves predicting a real-valued quantity. Below are some regression Loss Functions:-

● **Mean Squared Error Loss**

- The Mean Squared Error, or MSE, loss is the default loss to use for regression problems.
- Mean squared error is calculated as the average of the squared differences between the predicted and actual values
- loss function when compiling the model.
- It is recommended that the output layer has one node for the target variable and the linear activation function is used

● **Mean Absolute Error**

- The Mean Absolute Error, or MAE, loss is an appropriate loss function in this case as it is more robust to outliers.
- It is calculated as the average of the absolute difference between the actual and predicted values.
- e.g. large or small values far from the mean value

Binary

Classification

Loss

Functions

- The Binary classification are those predictive modelling problems where examples are assigned one of two labels.
- The problem is often framed as predicting a value of 0 or 1 for the first or second class and is often implemented as predicting the probability of the example belonging to class value 1.

Binary Classification Loss function are as Follows:-

- **Hinge**

- The hinge loss function has many extensions, often the subject of investigation with SVM models.
- It has the effect of smoothing the surface of the error function and making it numerically easier to work with.
- If using a hinge loss does result in better performance on a given binary classification problem, is likely that a squared hinge loss may be appropriate. As with using the hinge loss function.
- the target variable must be modified to have values in ~~test~~ $\{-1,1\}$.

- **Binary Cross Entropy/logsloss**

- Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1.
- It then calculates the score that penalizes the probabilities based on the distance from the expected value.
- That means how close or far from the actual value

Learning Rate

The learning rate is a hyper parameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas

a value too large may result in learning a sub-optimal set of weights too fast to run an unstable training process.

The learning rate may be the most important hyper parameter when configuring your neural network.

The amount that the weights are updated during training is referred to as the step size or the “learning rate.”

Specifically, the learning rate is a configurable hyper parameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.

The learning rate control shows quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

Critical

Scale:3Priority

Scale:1

System Design 3

Have a Train Section where we can specify the number of Epochs till which we can train the Model and the Batch Size.

Epochs

One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.



Figure 8: Train Section

Since one epoch is too big to feed to the computer at once we divide it in several smaller batches.

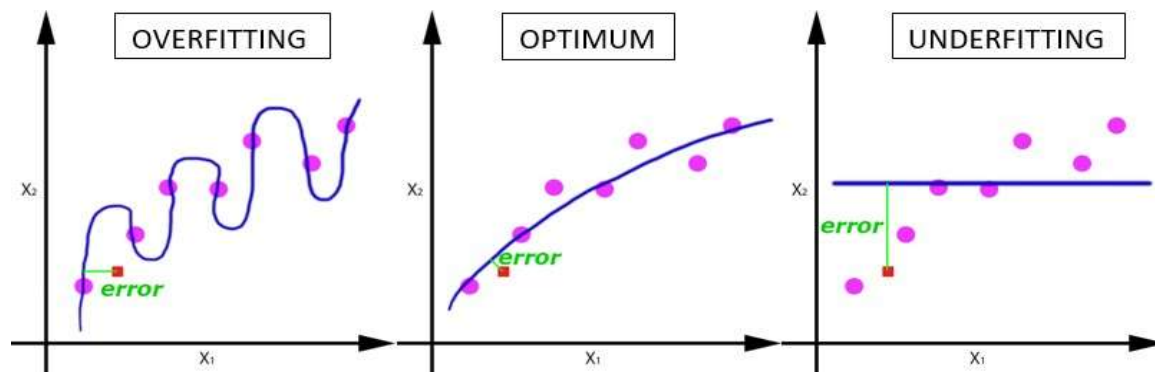


Figure 9

One epoch leads to under fitting of the curve in the graph (below).

As the number of epochs increases, more number of times the weight are changed in the neural network and the curve goes from under fitting to optimal to overfitting curve.

Batch Size

The batch size is a hyper parameter that defines the number of samples to work through before updating the internal model parameters.

Think of a batch as a for-loop iterating over one or more samples and making predictions. At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient.

A batch size of 32 means that 32 samples from the training dataset will be used to estimate the error gradient before the model weights are updated. One training epoch means that the learning algorithm has made one pass through the training dataset, where examples were separated into randomly selected “batch size” groups.

Batch size is typically chosen between 1 and a few hundreds, *e.g. [batch size] = 32 is a good default value*

The batch size impacts how quickly a model learns and the stability of the learning process. It is an important hyper parameter that should be well understood and tuned by the deep learning practitioner.

CriticalScale:3

PriorityScale:1

3.2 ARCHITECTURE DESIGN

In the Architectural designing of PyCoGen the user interacts with the UI/UX based web interface and also interacts with the each section named as follows Build, learn and train as shown in figure 10. The each individual section as given data by the user as an input and a choice in their respective fields and this input data is been feeded on to the visualization and algorithms as per the needs of the user for their models characteristics. The Visualizers and Algorithms are the machine learning library from Tensorflow.js and Keras an High API to instantiate the desired Hyper parameters from its configuration library. The provided input data to the Algorithms are processed and dynamic and rapid changes were made to the Generated Code as well as to the Model Architecture Preview where the visualization part takes place for the CNN.

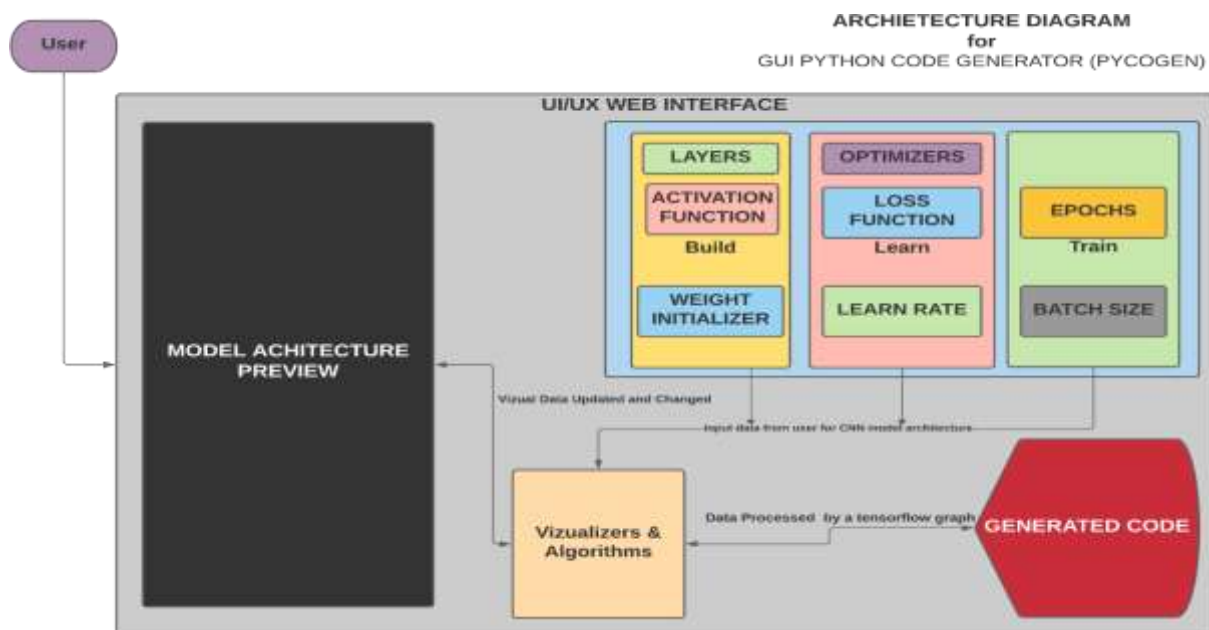
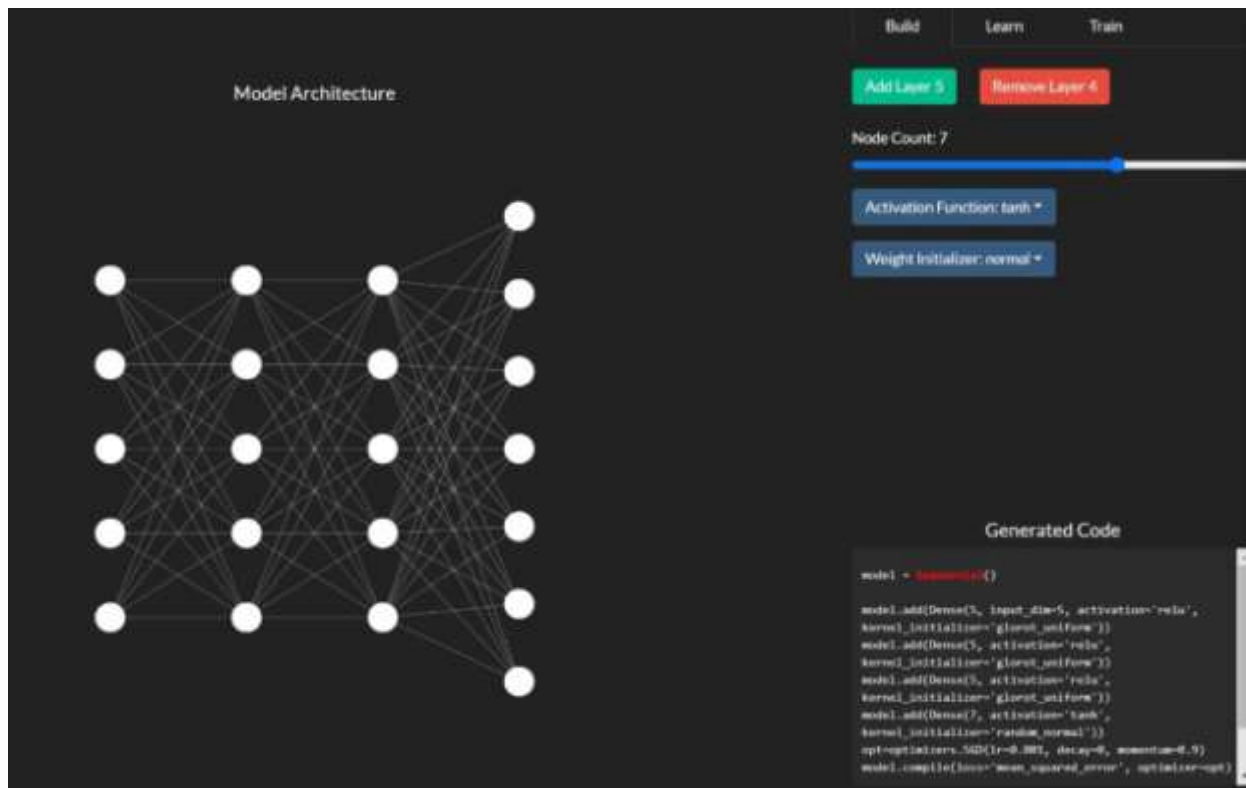


Figure 10: (ARCHITECTURE DESIGN OF GUI PYTHON CODE GENERATOR FOR CNN)

3.3 USER INTERFACE DESIGN

The product UI/UX Design enables the user to customize the architecture model by adding or removing the layers of the architecture layer in the build section of the product whereas the proportionate amount of number of nodes in each layer can be set using Node Count. To set the model architecture's Activation functions and weight of individual nodes there is a weight initializer and Activation Functions where user can set the desired parameters and final inference generated code is retrieved according to the parameters provided by the user

Figure 11:(functional and front-end view of the product is demonstrated)



CHAPTER-IV

CODING AND TESTING

4.1 SAMPLECODING

CODE FOR SYNTAX FORMATION AT THE GENERATED CODE OF PYCOGEN AT CODEPANEL.JS

```
classCodePanel extends Component {
  constructor(props) {
    super(props);
    this.state = {
      // currentCode: "model =Sequential()",
      kerasCode: newMap(),
      lossDict: new Map(),
      weightDict: newMap(),
    }
    this.state.kerasCode.set("beginModel", "model = Sequential()");
    this.state.kerasCode.set("addLayer", "model.add(Dense(");
    this.state.kerasCode.set("input_dim", "input_dim=");
    this.state.kerasCode.set("compile", "model.compile(");
    this.state.kerasCode.set("losser", "loss=");
    this.state.kerasCode.set("optimizer", "optimizer=");
    this.state.kerasCode.set("metrics", "metrics=[");
    this.state.kerasCode.set("modelFit",
      "model.fit(x=x_train,
      y=y_train,epochs=");
    this.state.kerasCode.set("batch", "batch_size=");
    this.state.kerasCode.set("learnRate", "learning_rate=");
    this.state.kerasCode.set("network_end", "));");
    this.state.kerasCode.set("loss", "loss=");
    this.state.kerasCode.set("learningRateDecay", "decay=");
    this.state.kerasCode.set("weightInit", "kernel_initializer=");
    this.state.lossDict.set("Mean Squared Error", "mean_squared_error");
    this.state.lossDict.set("Mean Absolute Error", "mean_absolute_error");
    this.state.lossDict.set("Hinge", "hinge");
    this.state.lossDict.set("Binary Crossentropy", "binary_crossentropy");
    this.state.weightDict.set("zeros", "zeros");
    this.state.weightDict.set("ones", "ones");
    this.state.weightDict.set("normal", "random_normal");
    this.state.weightDict.set("uniform", "random_uniform");
    this.state.weightDict.set("glorot uniform", "glorot_uniform");
    this.state.weightDict.set("he uniform", "he_uniform");
  }
  getActivationCode(layer) {
    return 'activation=' + "" + layer.activation + "";
  }
}
```

CODE FOR VIZUALIZATION AND WORKING OF BUILD SECTION IN BUILDPANEL.JS

```
classBuildPanel extends Component {
  constructor(props) {
    super(props);
    this.state = {
      test: 13,
      input: 12,
      value: 0,
      setValue: 0,
      currentActivation: "<select a layer>",
      currentInit: "<select a layer>",
      activations: [
        "linear", "relu", "tanh", "sigmoid", "softmax"
      ],
      inits: [
        "zeros", "ones", "normal", "uniform", "glorot uniform",
        "he uniform"
      ],
    };
    this.handleChange = this.handleChange.bind(this);
  }
  handleChange(event) {
    // this.props.appState.selectedLayer
    if (this.props.appState.loadable) {
      this.props.appState.doSetNumNodes(this.props.appState.selectedL
ayer, event.target.value);
      this.props.appState.hideModelPanel();
    }
  }
  getActivation(selectedLayer) {
    if (selectedLayer) return selectedLayer.activation;
    else return "";
  }
  getInit(selectedLayer) {
    if (selectedLayer) return selectedLayer.weightInit;
    else return "";
  }
  render() {
    const loadable = this.props.appState.loadable;
    constselectedLayer =
    this.props.appState.network.arrLayers[this.props.appState.selected
Layer];
  }
}
```

CODE FOR VIZUALIZATION AND WORKING OF ARCHITECTURE MODEL INMODEL.PANEL.JS

```
import { Sigma, LoadJSON } from 'react-sigma'
import { layer } from './kerasCode.js'
import './CodePanel.css'
functiongetMaxNodeCount(arrLayers) {
  let max = 0;
  for (let layer of arrLayers) {
    letnodeCount = layer.numNodes;
    if (nodeCount> max) max = nodeCount;
  }
  return max;
}
functionlayersToGraph(arrLayers, coloredLayer) {
```

```

let maxNodeCount = getMaxNodeCount(arrLayers);

let graph = {};

let nodes = [];

let edges = [];

let prevLayerNodes = []; // cache

for (let i = 0; i < arrLayers.length; i++) {

    let layer = arrLayers[i];

    let nodeCount = layer.numNodes;

    let x = i * 0.2;

    let color = i === coloredLayer ? "#add8e6" : "#fff";

    let vgap = 0.15 - nodeCount / 200;

    let renderHeight = (nodeCount - 1) * vgap;

    let initY = (2.0 - renderHeight) / 2;

    let thisLayerNodes = [];

    for (let j = 0; j < nodeCount; j++) {

        let y = initY + j * vgap;

        let id = i + "," + j;

        let size = 1;

        let node = { x: x, y: y, size: size, id: id, color: color };

        nodes.push(node);

        thisLayerNodes.push(node);

    }

    if (prevLayerNodes.length > 0) { // TODO check this??

        connect(prevLayerNodes, thisLayerNodes, edges);

```

```

    }

    prevLayerNodes = thisLayerNodes;

    }

    graph.nodes = nodes;

    graph.edges = edges;

    return graph;

}

function connect(prevLayerNodes, thisLayerNodes, edges) {

    for (let i = 0; i < prevLayerNodes.length; i++) {

        let prevNode = prevLayerNodes[i];

        let source = prevNode.id;

        for (let j = 0; j < thisLayerNodes.length; j++) {

            let thisNode = thisLayerNodes[j];

            let id = "e_" + prevNode.id + "," + thisNode.id;

            let target = thisNode.id;

            let size = 1 - (prevLayerNodes.length * thisLayerNodes.length) / 100;

            let edge = { id: id, source: source, target: target, size: size };

            edges.push(edge);

        }

    }

}

function writer(g) { // TODO

    let content = JSON.stringify(g);

    })

```

```

    }

    }

    export class ModelPanel extends Component {

    constructor(props) {

    super(props);

    }

    onClickNodeFunc = (e) => {

    const node = e.data.node;

    const id = node.id;

    const          SELECTED_LAYER
                    =parseInt(id.substring(0,id.indexOf(",")));

    this.props.appState.doSelectLayer(SELECTED_LAYER);

    }

    getSigma(data) {

    let s = <Sigma

    graph={data}

    style={{ height: "90%"

    }}onClickNode={this.onClickNodeFunc}settin

    gs={{

    maxNodeSize: 15, maxEdgeSize: 0.3,

    clone:false,

    enableHovering: true,

    rescaleIgnoreSize: false, // TODO change?

    }}>

    </Sigma>

    render() {

    let layers = this.props.appState.network.arrLayers

    let g = layersToGraph(layers, this.props.appState.coloredLayer)

    if (!this.props.appState.hiddenModelPanel) {

```

```

    return (

    <div id="content">

    <h2>PYCOGEN</h2>

    <h5 class="center">Model Architecture</h5>

    <div style={{ height: "100%"}}>

    {this.getSigma(g)}

    </div>

    </div>

    );

    } else{

    this.props.appState.unhideModelPanel(); // re-render!

    return false;

    }

    }

    }

```

CODE FOR VIZUALIZATION AND WORKING OF TRAIN SECTION IN TRAINPANEL.JS

```
import React, { Component } from "react";

import { Form } from "react-bootstrap";

class TrainPanel extends Component {

  constructor(props) {

    super(props);

    this.state = {

    };

    this.handleEpochChange = this.handleEpochChange.bind(this);
    this.handleBatchChange = this.handleBatchChange.bind(this);

    handleEpochChange(event) {

      this.props.appState.doSetEpochs(event.target.value)

    }

    handleBatchChange(event) {

      this.props.appState.doSetBatchSize(event.target.value)

    }

    render() {

      return (

        <div>

          <br/>

          <Form>

            <Form.Group controlId="formBasicRange">

              <Form.Label>Epochs:
              <i>{this.props.appState.network.epochs}</i></Form.Label>

              <Form.Control type="range"

                onChange={this.handleEpochChange}

                max="500"

                value={this.props.appState.network.epochs}

              />

            </Form.Group>

          </Form>

          <Form>

            <Form.Group controlId="formBasicRange">

              <Form.Label>Batch Size:
              <i>{this.props.appState.network.batchSize}</i></Form.Label>

              <Form.Control type="range"

                onChange={this.handleBatchChange}

                min="32"

                max="256"

                step="32" value={this.props.appState.network.batchSize}

              />

            </Form.Group>

          </Form>

        </div>

      );

    }

  }

}

export default TrainPanel;
```

CODE FOR VIZUALIZATION AND WORKING OF LEARN SECTION IN LEARNPANEL.JS

```
class LearnPanel extends Component {
  constructor(props) {
    super(props);
    this.state = {
      test: 13,
      input: 12,
      value: 0,
      setValue: 0,
      currentOpti: "SGD",
      optimizers: [
        "SGD", "RMSProp", "Adadelta", "Adam"
      ],
      losses: [
        "Mean Squared Error", "Mean Absolute Error", "Hinge",
        "Binary Crossentropy"
      ],
    }

    this.handleChange = this.handleChange.bind(this);
    this.handleClick = this.handleClick.bind(this);

    handleChange(event) {

      if (this.props.appState.loadable) {
        this.props.appState.doSetLearnRate(event.target.value)
      }
    }

    handleClick(event){
      if(this.props.appState.network.learningRateDecay){
        this.props.appState.doSetLearningDecay(false);
      }
      else{
        this.props.appState.doSetLearningDecay(true);
      }
    }

    render() {
      return (
        <div>
          <br/>
          <Dropdown>
            <Dropdown.Toggle variant="primary" id="dropdown-basic">Optimizer:
            <i>{this.state.currentOpti}</i></Dropdown.Toggle>

            <Dropdown.Menu>
              {this.state.optimizers.map(opti => {
                return(
                  <Dropdown.Item as="button"
                    onClick={() => {
                      this.setState({currentOpti: opti});
                      this.props.appState.doSetOptimizer(opti);
                    }}
                    >{opti}</Dropdown.Item>
                )
              })}
          </Dropdown>
        </div>
      )
    }
  }
}
```

```

</Dropdown.Menu>

</Dropdown>

<p></p>

<Dropdown>

<Dropdown.Toggle variant="primary" id="dropdown-basic">Loss
Function:
<i>{this.props.appState.network.loss}</i></Dropdown.Toggle>

<Dropdown.Menu>

      {this.state.losses.map(loss => {

return(

<Dropdown.Item as="button"

onClick={() => {

          // this.setState({currentOpti: loss});

this.props.appState.doSetLoss(loss);

        }}

>{loss}</Dropdown.Item>

      )

    }}}

</Dropdown.Menu>

</Dropdown>

<br/>

<Form>

<Form.Group controlId="formBasicRange">

<Form.Label>Learning rate of the network:
{this.props.appState.loadable
?this.props.appState.network.learnRate : ""}</Form.Label>

<Form.Control type="range"

onChange={this.handleChange}

max="0.1"

step="0.001"

```

```

value={this.props.appState.network.learnRate} />

</Form.Group>

</Form>

<Form>

      {['checkbox'].map((type) => (

<div key={'learning-rate-decay'} className="mb-3">

<Form.Check

type={type}

id={`learning rate checkbox`}

label={`Learning Rate Decay`}

onChange={this.handleClick}

          //

checked={this.props.appState.network.learningRateDecay}

        />

</div>

      )))

</Form>

</div>

);

}

}

export default LearnPanel;

```

4.2 TEST STRATEGY AND APPROACHES

Testing may be unit testing(generally preferred) , end to end testing , integration testing etc

Jest is a JavaScript testrunner(Unit Testing) that lets you access the DOM via jsdom. While jsdom is only an approximation of how the browser works, it is often good enough for testing React components. Jest provides a great iteration speed combined with powerful features like mocking modules and timers so you can have more control over how the code executes.

The testing screenshots are as follows:

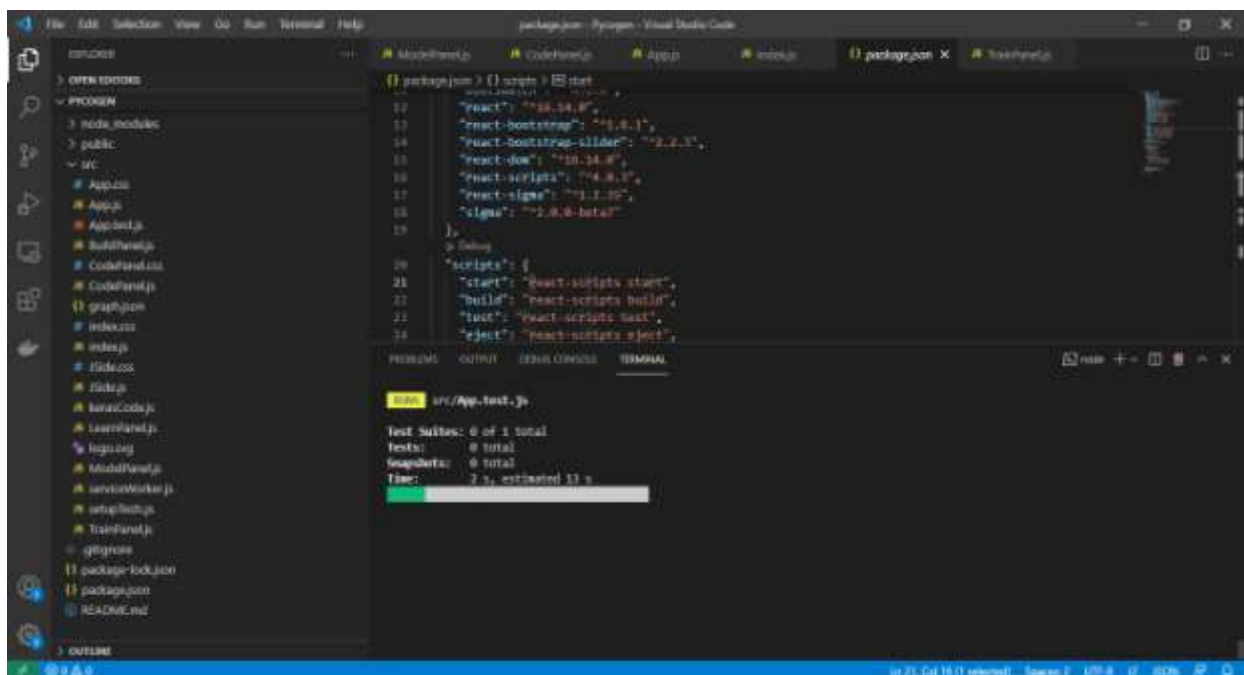


Figure 12: Unit testing of PyCoGen by JEST

PASS src/App.test.js					
App					
✓ renders a welcome view (6ms)					
File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	100	100	100	100	
App.js	100	100	100	100	

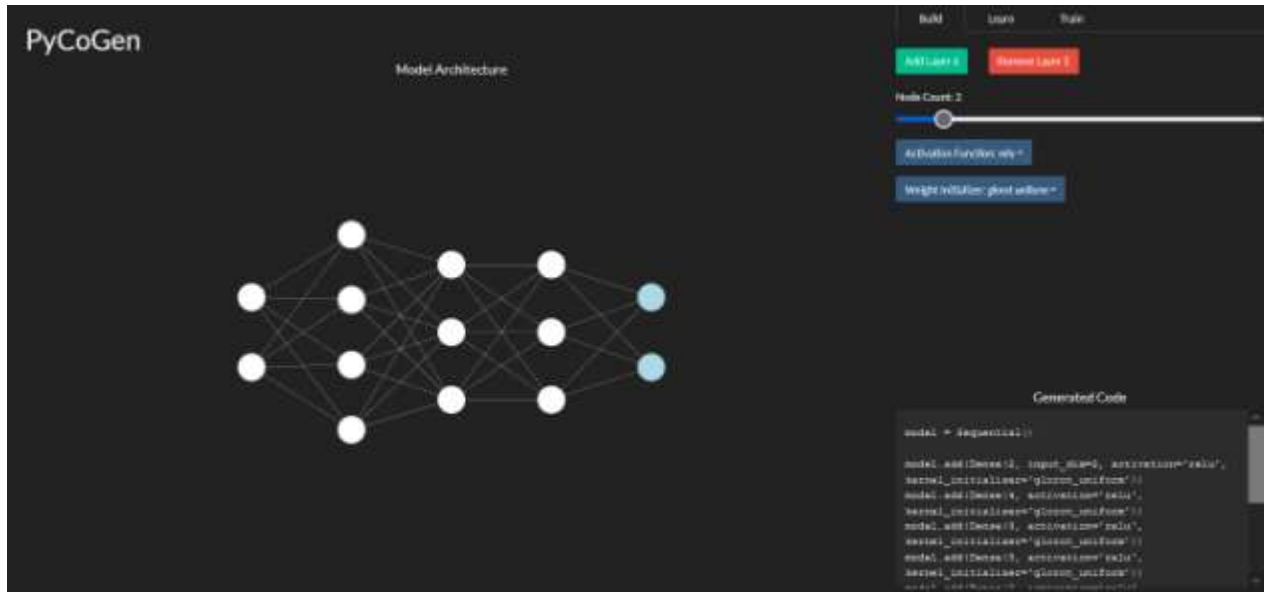
Figure 13: All tests successfully passed

Chapter V

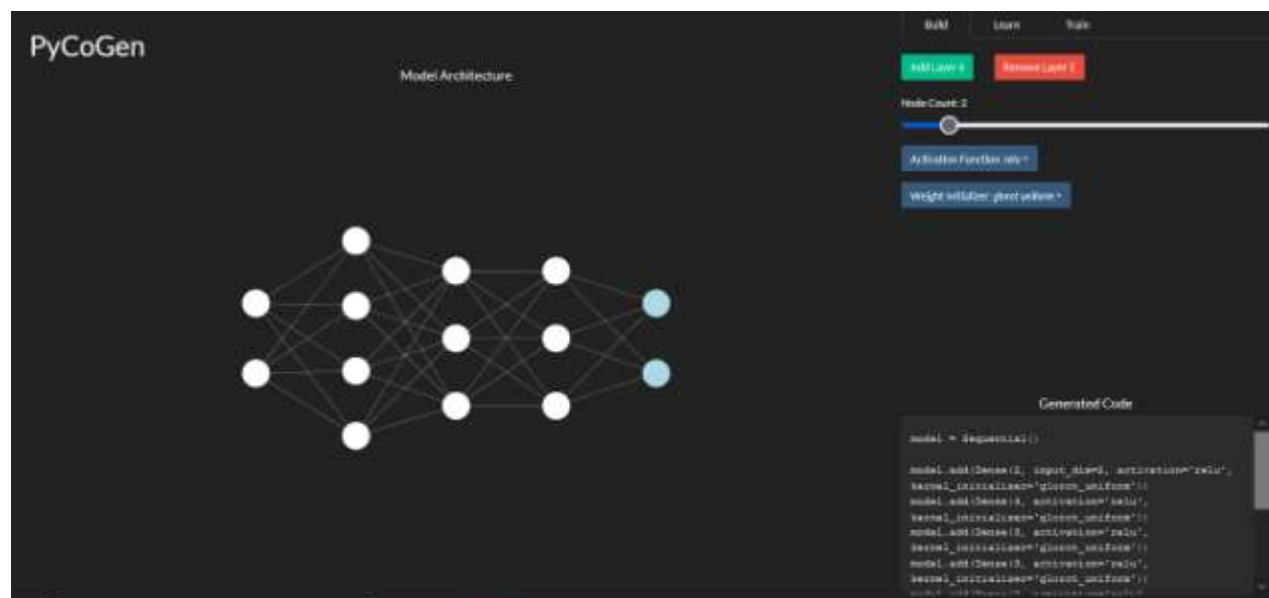
RESULTS AND DISCUSSION

5.1 Product Screenshots

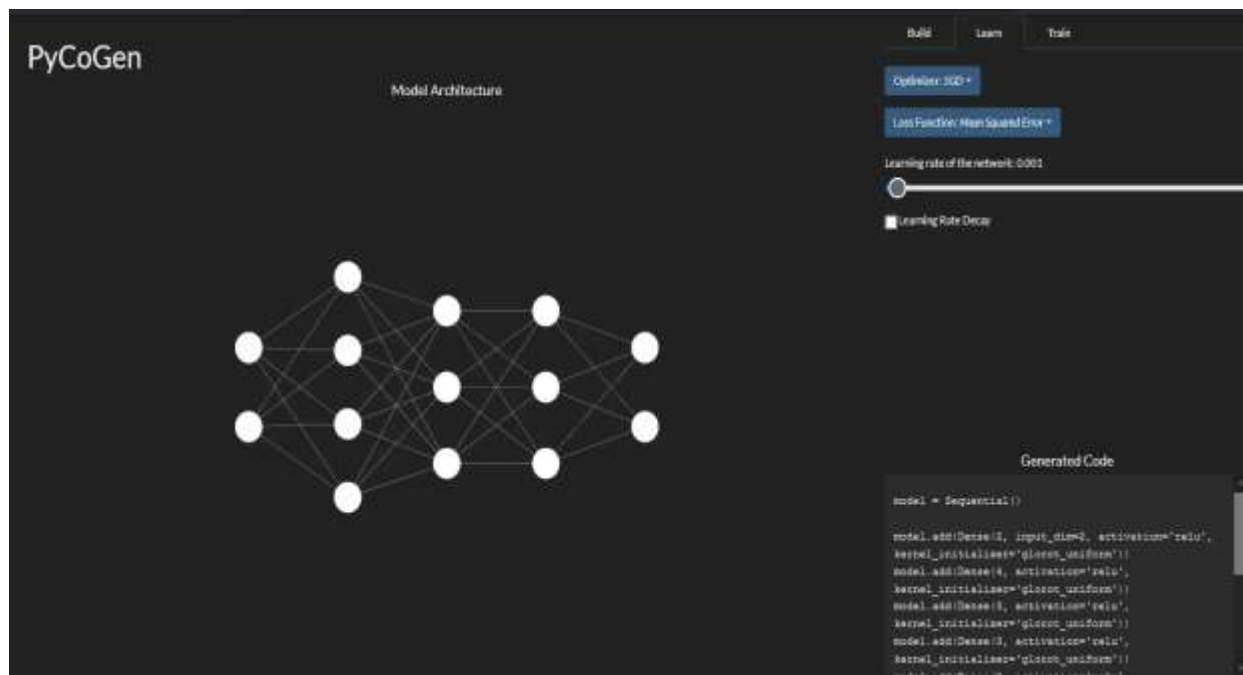
The below shown screenshots gives the resulting final view and working of GUI Python Code Generator for CNN (PyCoGen)



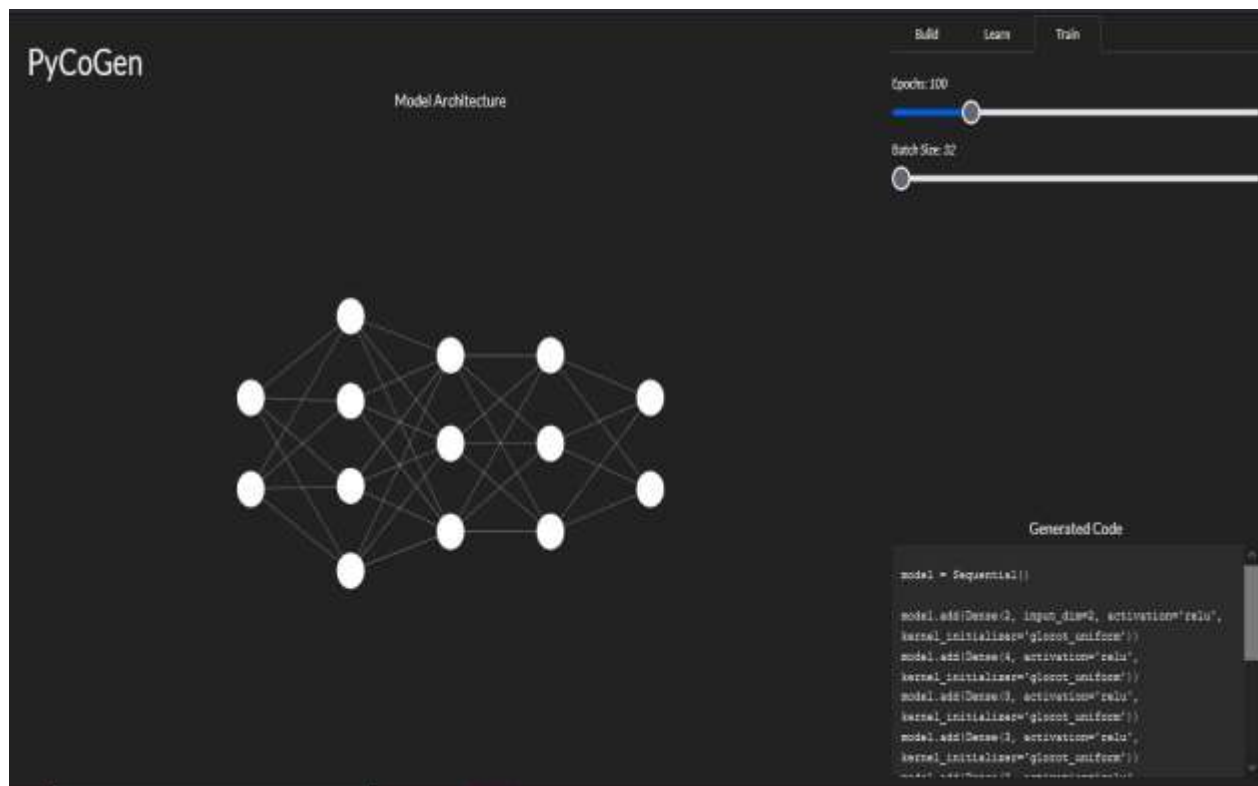
Screenshot1



Screenshot2

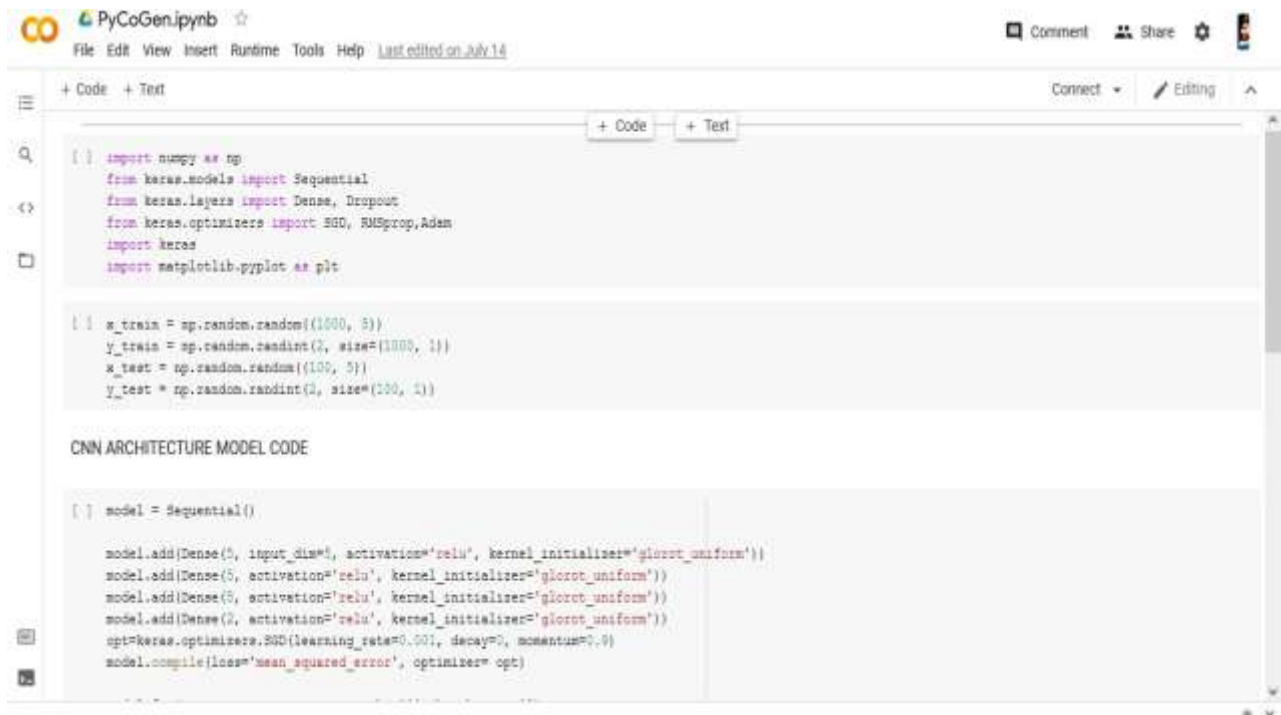


Screenshot3



Screenshot4

Demonstration of the Working code generated by PyCoGen on a dataset (Google Colaboratory) via screenshots



The screenshot shows a Google Colaboratory notebook titled "PyCoGen.ipynb". The code is written in Python and includes imports for numpy, keras, and matplotlib. It defines training and testing data sets and a CNN architecture model.

```
[ ] import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import SGD, RMSprop, Adam
import keras
import matplotlib.pyplot as plt

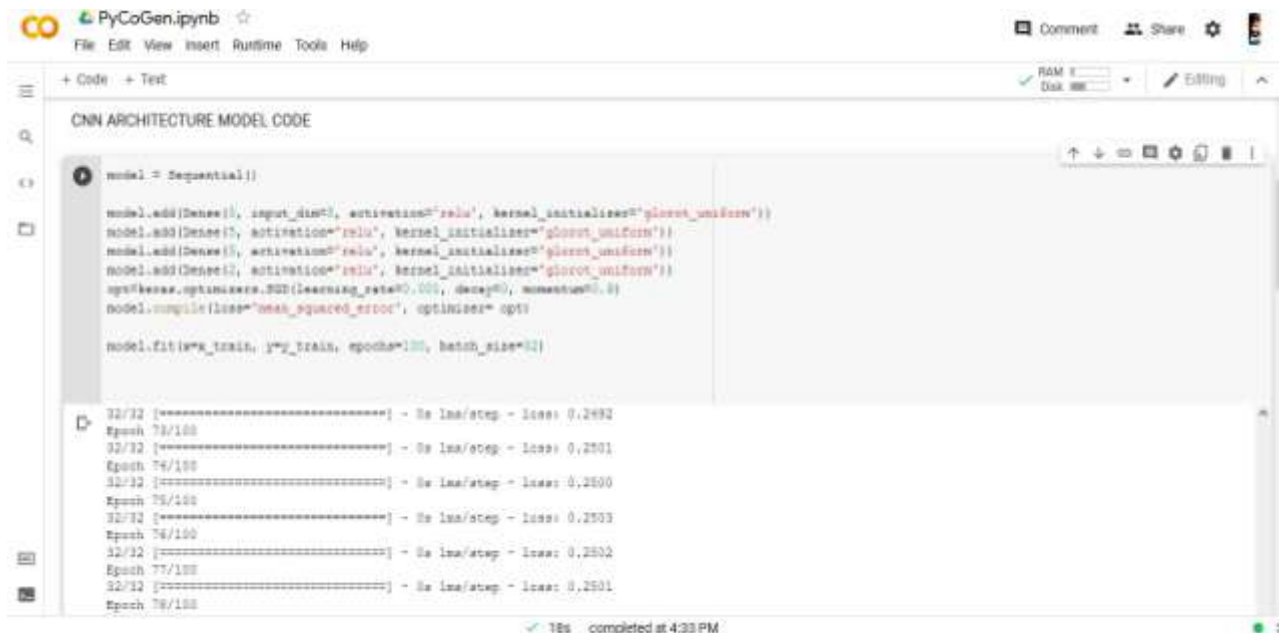
[ ] x_train = np.random.random((1000, 5))
y_train = np.random.randint(2, size=(1000, 1))
x_test = np.random.random((100, 5))
y_test = np.random.randint(2, size=(100, 1))

CNN ARCHITECTURE MODEL CODE

[ ] model = Sequential()

model.add(Dense(5, input_dim=5, activation='relu', kernel_initializer='glorot_uniform'))
model.add(Dense(5, activation='relu', kernel_initializer='glorot_uniform'))
model.add(Dense(5, activation='relu', kernel_initializer='glorot_uniform'))
model.add(Dense(2, activation='relu', kernel_initializer='glorot_uniform'))
opt=keras.optimizers.SGD(learning_rate=0.001, decay=0, momentum=0.9)
model.compile(loss='mean_squared_error', optimizer= opt)
```

Screenshot 5: demonstration of working mode code on a data set



The screenshot shows the same Google Colaboratory notebook, but now the code is executed. The output shows the model's performance over 100 epochs, with the loss decreasing from 0.2493 to 0.2501.

```
CNN ARCHITECTURE MODEL CODE

[ ] model = Sequential()

model.add(Dense(5, input_dim=5, activation='relu', kernel_initializer='glorot_uniform'))
model.add(Dense(5, activation='relu', kernel_initializer='glorot_uniform'))
model.add(Dense(5, activation='relu', kernel_initializer='glorot_uniform'))
model.add(Dense(2, activation='relu', kernel_initializer='glorot_uniform'))
opt=keras.optimizers.SGD(learning_rate=0.001, decay=0, momentum=0.9)
model.compile(loss='mean_squared_error', optimizer= opt)

model.fit(x_train, y_train, epochs=100, batch_size=32)

32/32 [=====] - 3s 1ms/step - loss: 0.2493
Epoch 79/100
32/32 [=====] - 3s 1ms/step - loss: 0.2501
Epoch 76/100
32/32 [=====] - 3s 1ms/step - loss: 0.2500
Epoch 75/100
32/32 [=====] - 3s 1ms/step - loss: 0.2503
Epoch 74/100
32/32 [=====] - 3s 1ms/step - loss: 0.2502
Epoch 77/100
32/32 [=====] - 3s 1ms/step - loss: 0.2501
Epoch 78/100
```

18s completed at 4:33 PM

Screenshot 6: generated model code Successfully working on a data set

The screenshot shows the PyCoGen.ipynb interface with the following content:

```
[3] Epoch 99/100  
32/32 [=====] - 0s 1ms/step - loss: 0.2500  
Epoch 100/100  
32/32 [=====] - 0s 1ms/step - loss: 0.2506  
<keras.callbacks.History at 0x7f506c7e4d80>
```

Below the code, the model summary is displayed:

```
model.summary()  
test_acc = model.evaluate(x_test, y_test, verbose=2)  
print(test_acc)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 5)	30
dense_1 (Dense)	(None, 5)	30
dense_2 (Dense)	(None, 5)	30
dense_3 (Dense)	(None, 2)	12

Total params: 102
Trainable params: 102
Non-trainable params: 0

4/4 - 0s - loss: 0.2507
0.25071075558462413

At the bottom, it says "completed at 4:34 PM".

Screenshot 7: model summary and test accuracy of model code generated by PyCoGen

PYCOGEN is a tool which is for deep learning implementations and provides a GUI based interactive interface to the user for visualization of nodes and layers and for defining the hyper parameters from a list of available options corresponding to which CNN Architecture base model code in python is generated whereas in existing technologies like Cog is just a simple code generation tool written in python and generates a simple python code, Pixtocode uses UML diagram as an input from user for which the code is generated. It specifically focuses to generate code for a websites and Blockly has an interactive GUI to generate code in an interactive puzzle like format but it requires users skills and knowledge of the basic programming languages and approaches. Since all the existing technologies focuses on automatic generation of code for programing languages, PyCoGen stands out as a standalone tool for advance deep learning Convolutional neural network which neither of the existing tools qualify for their expertise.

Chapter VI

CONCLUSION AND SCOPE

6.1 Conclusion

GUI Python Code Generator for CNN(PyCoGen) falls under the category of fully developed computer-assisted software engineering (CASE) tools. The Code Generation for CNN architecture base model has been fully functional and automated. The web app is also hosted on a CI/CL server popularly recognized as netlify.com. The link is given below

[PyCoGen \(Click Here\)](#)

This app is purposely build to make a learning environment and to give a head on start in Neural networks to all those deep learning, data science enthusiast, scholars, etc. We Hope PyCoGen creates a convenient work/learning environment for all of us.

6.2 Future Scope

GUI Python Code Generator(PYCOGEN) in-browser GUI Code Generator tool which is a sub-system for generating python code with version 1.0 approved. This code generators can further be used and implemented as a standalone tools on various Notebooks, Editors and IDE (for e.g. Anaconda, etc) and can be utilized for project to be targeted to improve or add more features to it (design and code the application – it sets the guidelines for future development) that are focused on deep learning and even for more critical and complex neural network. The specifics of the tools is to provide a hassle free low and easy coding Environment for complex problems. CNN Introduces tons of Complexity when it's a matter of creating a model and architecture and that's where the tools comes in and plays a vital role in saving time and focus of the user on the work they're up to, also it cuts down the cost in relevance to the cost estimation namely Line of code(LoC) to the End-User with further research and study and the use of appropriate technologies and tools, we look forward to improving the features of the product and bringing it to a higher level.

References

A. W. Harley. (2015). "An Interactive Node-Link Visualization of Convolutional Neural Networks," *ISVC*, 267-277.

<https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size>. (n.d.).

<https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/><https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>.

<https://towardsdatascience.com/>

<https://towardsdatascience.com/overview-of-various-optimizers-in-neural-networks-https://www.cs.ryerson.ca/~aharley/vis/>.(n.d.). Retrieved from <https://www.cs.ryerson.ca/~aharley/vis/>

<https://www.freecodecamp.org/news/tensorspace-js-a-way-to-3d-visualize-neural-networks-in-browsers-2c0afd7648a8/>.

https://www.tutorialspoint.com/nodejs/nodejs_environment_setup.htm.(n.d.). Retrieved from https://www.tutorialspoint.com/nodejs/nodejs_environment_setup.htm:

Sethi, A. (2017). DLPaper2Code: Auto-generation of Code from Deep Learning Research Papers.*arXiv:1711.03543v1 [cs.LG]*, 8.

visualizing-artificial-neural-networks-anns. (2020). Retrieved from [datascience.com](https://towardsdatascience.com/visualizing-artificial-neural-networks-anns): •

www.frontiersin.org/.(n.d.). Retrieved from [www.frontiersin.org/https://www.frontiersin.org/articles/10.3389/frai.2020.00049/full](https://www.frontiersin.org/articles/10.3389/frai.2020.00049/full)

End of the Document.