



**CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)**

Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University, Chandigarh

Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943

Website: [www.ccet.ac.in](http://www.ccet.ac.in) | Email: [principal@ccet.ac.in](mailto:principal@ccet.ac.in) | Fax. No. :0172-2750872



---

## Department of Computer Sc. & Engineering

---

### Summer Training Report

ON

### WEB DEVELOPMENT

A Project Report Submitted in Partial Fulfilment  
of the requirements for the award of

**Bachelor of Engineering**

in

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**SIDDHARTH SAMBER**

(CO17358)

Under the supervision of

**(Mr. Daniel Randall)**



**CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE  
WING)**

(Government Institute Under UT Administration | Affiliated to Punjab University, Chandigarh)

Sector 26, Chandigarh, PIN-160019

JUNE, 2020



---

## Department of Computer Sc. & Engineering

---

### CANDIDATE DECLARATION

This is declare that the work presented in this report entitled “**Neural Network Visualization**”, in the fulfilment of the requirement for the award of the Bachelor of Engineering in Computer Sciences & Engineering, submitted in CSE Department, Chandigarh Collage of Engineering & Technology (Degree Wing) affiliated to Punjab University Chandigarh is an authentic record of my/our work carried out during my degree under the guidance of **Mr. Daniel Randall (Coursera)**. The work reported in this has not submitted by me for award of any other degree or diploma.

Date : August 2, 2020

Siddharth Samber

Place: Coursera (Online)

CO17358



07/19/2020

**Siddharth Samber**

has successfully completed

**Create Your First Web App with Python and Flask**

an online non-credit course authorized by Coursera Project Network and offered through Coursera

Amit Yadav  
Instructor  
Machine Learning and Data Science

**COURSE  
CERTIFICATE**



Verify at [coursera.org/verify/F3VNGJ8QBZM9](https://coursera.org/verify/F3VNGJ8QBZM9)  
Coursera has confirmed the identity of this individual and their participation in the course.



07/19/2020

**Siddharth Samber**

has successfully completed

**Neural Network Visualizer Web App with Python**

an online non-credit course authorized by Coursera Project Network and offered through Coursera

Amit Yadav  
Instructor  
Machine Learning and Data Science

**COURSE  
CERTIFICATE**



Verify at [coursera.org/verify/VU9QPBCYLYZH](https://coursera.org/verify/VU9QPBCYLYZH)  
Coursera has confirmed the identity of this individual and their participation in the course.



07/27/2020

**Siddharth Samber**

has successfully completed

**Programming for Everybody (Getting Started with Python)**

an online non-credit course authorized by University of Michigan and offered through Coursera

Charles Severance  
Clinical Professor, School of Information  
University of Michigan

**COURSE  
CERTIFICATE**



Verify at [coursera.org/verify/P3ER7HKDzEDU](https://coursera.org/verify/P3ER7HKDzEDU)  
Coursera has confirmed the identity of this individual and  
their participation in the course.



08/02/2020

**Siddharth Samber**

has successfully completed

**Introduction to Web Development**

an online non-credit course authorized by University of California, Davis and offered through Coursera

Daniel Randall  
Web Development Instructor  
UC Davis Division of Continuing and Professional Education

**COURSE  
CERTIFICATE**



Verify at [coursera.org/verify/FC57RFER4KFJ](https://coursera.org/verify/FC57RFER4KFJ)  
Coursera has confirmed the identity of this individual and  
their participation in the course.



---

## Department of Computer Sc. & Engineering

---

### ACKNOWLEDGEMENT

I would like to express My whole hearted gratitude towards my esteemed of **Mr. Daniel Randall** (Coursera India) for his able guidance, monitoring and encouraging attitude and searching supervision throughout my study which enable me to successfully complete my project. Without his help and valuable suggestion, this report would not have been the light of the day. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I would also like to express a deep sense of gratitude to **Dr. Sunil. K Singh (Head Of Department C.S.E, CCET)** and **Dr. Ankit Gupta (Assistant Professor, C.S.E, CCET)** for their cordial support, valuable information and guidance which helped me in completing this task through various stages.

Lastly, I would like to thank almighty and my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.

Siddharth Samber

CO17358

## **ABSTRACT**

The Neural network with 1 input layer , 2 hidden layer, 1 ouptut layer will be created .We will visulaize the output values of all these layers of all the nodes.Every node value will be represented with a grayscale color depicting its value. We will create server with the help of flask to serve our Neural network Model and web application with the help of Streamlit and to obtain output of hidden layers we will use keras functional API.We will train our model with the help of keras. Dataset that we will use is EMNIST dataset, it is a dataset of handwritted digits of size 28X28. We will get a input image from our dataset and then we will try to predict it by clicking on the prediction button. And the output will be visual representation of all the layers in our model and also the output layer, which will be shown tells what is our output. It basically helps us to visualize our neural network.We can also view our inputted image from sidebar.

## TABLE OF CONTENTS

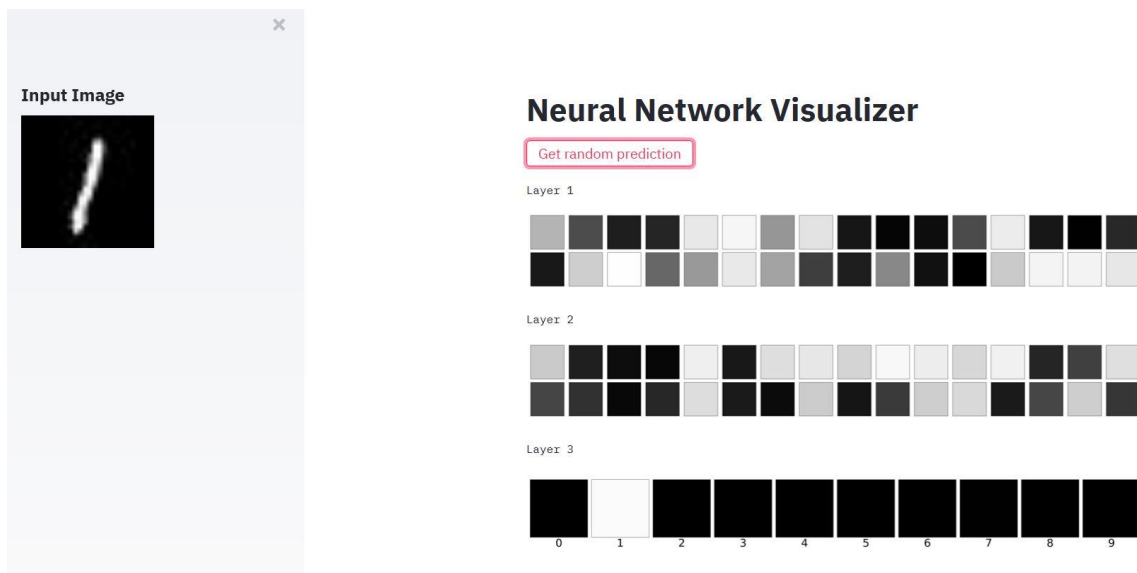
<b>S. No.</b>	<b>CHAPTERS</b>	<b>Page no.</b>
1	<b>Student Declaration</b>	2
2	<b>Certification by the Guide</b>	3-4
3	<b>Acknowledgement</b>	5
4	<b>Abstract</b>	6
5	<b>Table Of Contents</b>	7
6	<b>Chapter 1 HOW OUR PROJECT WORKS , TECHNOLOGIES USED</b>	8-9
7	<b>Chapter 2 DATASET AND MODEL</b>	10-11
8	<b>Chapter 3 SETTING ENVIRONMENT, ACTIVATION AND INSTALLATIONS</b>	12-14
9	<b>Chapter 4 FLASK</b>	15-18
10	<b>Chapter 5 Jinjer Templating</b>	19-23
11	<b>Chapter 6 JSON AND REST CALL</b>	24-28
12	<b>Chapter 7 STREAMLIT</b>	29
13	<b>Chapter 8 NUERAL NETWORKS</b>	30-35
14	<b>Chapter 9 CODE SNIPPETS AND EXPLANATION</b>	36
15	<b>Chapter 10</b>	
16	<b>Conclusion</b>	
17	<b>Future Scope</b>	
18	<b>Bibliography</b>	

## CHAPTER - 1

### HOW OUR PROJECT WORKS , TECHNOLOGIES USED

#### HOW OUR PROJECT WORKS

The Neural network with 1 input layer , 2 hidden layer, 1 output layer will be created .We will visualize the output values of all these layers of all the nodes. Every node value will be represented with a grayscale color depicting its value. We will create server with the help of flask to serve our Neural network Model and web application with the help of Streamlit and to obtain output of hidden layers we will use keras functional API. We will train our model with the help of keras. Dataset that we will use is EMNIST dataset, it is a dataset of handwritten digits of size 28X28. We will get a input image from our dataset and then we will try to predict it by clicking on the prediction button. And the output will be visual representation of all the layers in our model and also the output layer, which will be shown tells what is our output. It basically helps us to visualize our neural network. We can also view our inputted image from sidebar.



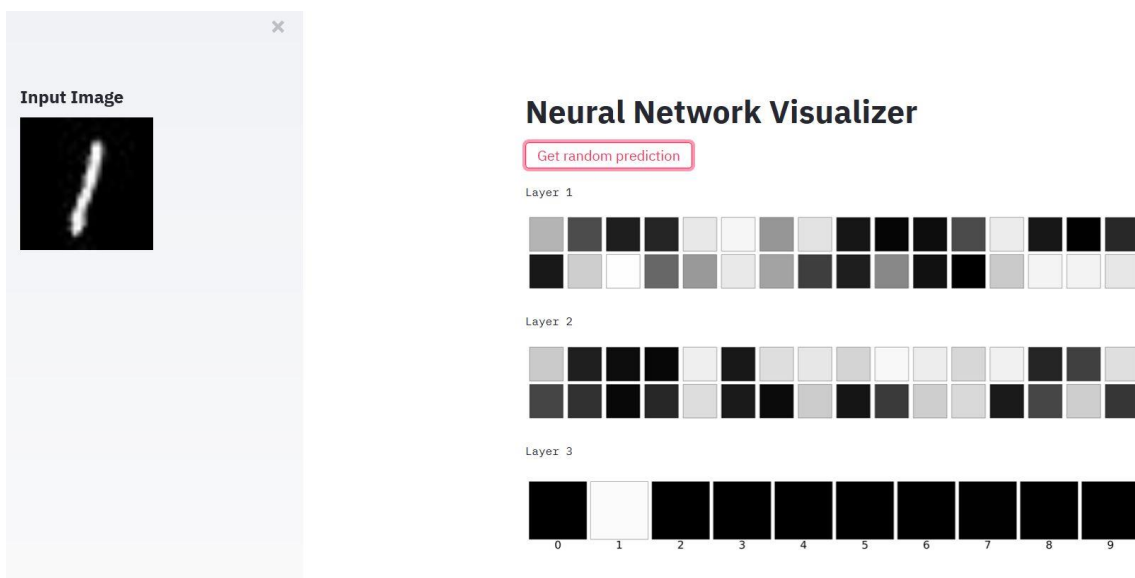
#### OVERVIEW OF TECHNOLOGIES USED

- Python :- We will use Python as our programming language . It is an interpreted, high-level, general-purpose programming language.
- Streamlit :- It is a open-source app framework which make it easy for data scientists and machine learning engineers to create high performance apps easily. We will use this to create our web application.



- **Flask :-** Flask is a micro web framework written in Python. We will use this to create our model Server.
- **Keras:-** Keras is an open-source neural-network library written in Python. We will use this to create our Deep learning model.
- **JSON:-** We will use JSON as our data interchange format.
- We will use keras functional API so that output of all the layers can be visualized
- We will use REST kind a software architectural style.

Libraries and modules used:- we have used Flask, Tensorflow, json, streamlit, numpy, random, Matplotlib, keras, requests etc.

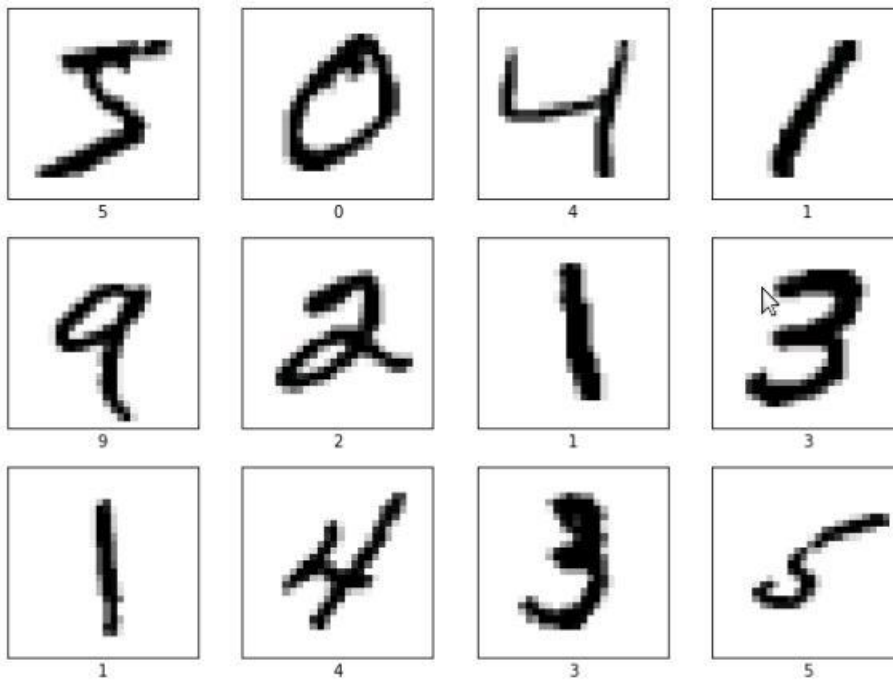


## CHAPTER -2

### DATASET AND MODEL

#### DATASET

Our Dataset consist of 70,000 Grayscale images of handwritten digits. Size of each image is  $28 \times 28$ . As it is grayscale so we only have one channel. We have used 60,000 images for training and 10,000 images for testing.



Preprocessing: we have unroll the  $28 \times 28$  dimension of an image to a vector of size 784 i.e single dimension array with size 784( $28 \times 28$ ). After that we have Normalized our dataset scaling the value which are between (0-255) to (0-1), which will lead to better convergence of our training model.

#### MODEL

##### Layers in our Model:

1. Input layer has 784 nodes
2. First dense input layer has 32 nodes and activation function used is sigmoid.
3. Second dense input layer has 32 nodes and activation function used is sigmoid

4. Output dense layer has 32 nodes and activation function used is softmax as it given probalastic distribution of each outcome and does classification easily

**Model specifications:**

- Loss function used is sparse categorical cross entropy sparse as we have not hot encoded our lables
- Optimizer used is adam
- Epochs are set to 30
- Batch size is sdet to 1024
- Verbose is set to 2

We have achieved 95% accuracy.

```
Epoch 26/30
59/59 - 0s - loss: 0.1291 - accuracy: 0.9646 - val_loss: 0.1515 - val_accuracy:
0.9553
Epoch 27/30
59/59 - 0s - loss: 0.1262 - accuracy: 0.9657 - val_loss: 0.1490 - val_accuracy:
0.9559
Epoch 28/30
59/59 - 0s - loss: 0.1231 - accuracy: 0.9661 - val_loss: 0.1476 - val_accuracy:
0.9558
Epoch 29/30
59/59 - 0s - loss: 0.1202 - accuracy: 0.9674 - val_loss: 0.1459 - val_accuracy:
0.9568
Epoch 30/30
59/59 - 0s - loss: 0.1175 - accuracy: 0.9681 - val_loss: 0.1433 - val_accuracy:
0.9575
```

## **CHAPTER -3**

### **SETTING ENVIRONMENT, ACTIVATION AND INSTALLATIONS**

#### **Setting up Virtual Environments**

To start working with the project, we have to do some basic set up so that we can proceed smoothly. Create a virtual environment so that you do not mess up your existing python libraries and functions on your system.

#### **What is virtual environment**

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. This is one of the most important tools that most of the Python developers use.

#### **Why virtual environment**

Virtual Environment should be used whenever you work on any Python based project. It is generally good to have one new virtual environment for every Python based project you work on. So the dependencies of every project are isolated from the system and each other.

Let's take a scenario where you are working on two web based python projects and one of them uses a Django 1.9 and the other uses Django 1.10 and so on. In such situations virtual environment can be really useful to maintain dependencies of both the projects.

#### **How does virtual environment work**

We use a module named virtualenv which is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executables to use the packages that a Python project would need.

Install virtualenv

```
pip install virtualenv
```

```
virtualenv --version
```

#### **Create Virtual environment**

virtualenv can be created using the following command:

```
virtualenv my_name
```

After running this command, a directory named my\_name will be created. This is the directory which contains all the necessary executables to use the packages that a Python project would need. This is where Python packages will be installed.

## Activating Virtual Environment

Now after creating virtual environment, we need to activate it. Remember to activate the relevant virtual environment every time you work on the project. This can be done using the following command:

```
source virtualenv_name/bin/activate
```

Once the virtual environment is activated, the name of your virtual environment will appear on left side of terminal. This will let you know that the virtual environment is currently active

```
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL\Documents\flask>env\Scripts\activate

(env) C:\Users\DELL\Documents\flask>python server.py
```

## Installing Flask and Run python script

We can install flask in our Environemnt by the following command:

```
pip install flask
```

To run a Python script store in a '.py' file in command line, we have to write 'python' keyword before the file name in the command prompt.


```
python server.py
```

```
* Serving Flask app "server" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
```

## Installing Streamlit and Run Streamlit

Streamlit can be installed with pip, the Python package manager. It's a good idea to install Streamlit into a virtual environment, rather than installing Streamlit into the base or built-in version of Python installed on your machine. And to install streamlit we just have to use simple command `pip install streamlit`.

After the file is saved, simple your python file can be run as a Streamlit app from the command line using the following command.

 C:\Windows\System32\cmd.exe - streamlit run app.py

```
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\DELL\Documents\flask>env\Scripts\activate

(env) C:\Users\DELL\Documents\flask>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.4:8501
```

## CHAPTER – 4 (FLASK)

### INTRODUCTION

Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. Flask is also extensible and doesn't force a particular directory structure or require complicated boilerplate code before getting started.

Flask uses the Jinja template engine to dynamically build HTML pages using familiar Python concepts such as variables, loops, lists, and so on.

Flask is more flexible compared to other Python web frameworks like Django, but the onus is on you to get things working correctly as there are not many rules or “Flask way” of doing things.

The microframework in case of flask does not mean it can only do little or provides less functionality. It means it is more extensible and gives you the freedom to choose and decide the custom extensions and plugins to use.

### Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

The very first line imports the Flask from its library, and the next line creates a variable name and instantiates the app using Flask class.

The next line `@app.route('/') (/ means the home page )` is called a route which is used to redirect the user to a specific page or perform a particular function when the user visit's a particular URL. You can create your routes and specify the things functions should do when particular routes are called.

### Routes in Flask

Routes are places on a website that the users visit. Its also commonly known as URL (Uniform Resource Locator). It is important to have meaningful URL's to make it possible for users to remember and revisit the website easily. It also helps in SEO.

Flask uses the method route() to bind functions to a URL. Whenever a user visits a URL, the method attached to it executes.

You can create a custom URL with angle brackets (<>) and colon (:) as illustrated below.

```
@app.route("/user/<name>")
def display_user(name):
    # A string of any length(without slashes) can be assigned to the variable
    print(name)
```

### Accessing URL using method names

You can use url\_for() to easily access the route or a method attached to the route. The first argument is function name, and you can pass other arguments of any length which comply with variable rules of the route. A simple example below illustrates the method url\_for

```
@app.route("/login/<user>")
def login_user(user):
    # User login flow.
def redirect_to_login():
    url_for('login_user',user='some_user_name')
```

### Handling Requests

When a user visits a website or makes a request, the website has to respond appropriately. Flask supports both GET and POST request. By default Flask only accepts GET request. In order to accept post requests, it has to be specified in parameters of the function route()

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

### Static Files and templates.



Every modern web page needs CSS and JavaScript. To use the CSS and JS in Flask, you need to create a static folder in the working directory of your flask application. You can place all your CSS and JS files there and access them in your project.

Templates folder in Flask is used to store HTML files and other web pages you want to display with the application. When you refer to the web page in your code, it will look into the folder templates in the project directory.

Example Directory Structure:

```
|----- Project/
|--- flask_app.py
|--- static/
|   |-- style.css
|   |-- script.js
|--- templates/
|   |-- base.html
|   |-- home.html
|   |-- login.html
```

## Rendering templates

You can render templates in the flask using the method `render_template`.

```
from Flask import render_template

@app.route("/")
def display_home():
    return render_template('home.html', thing_to_say='hello')
```

## Reading data from Forms and working with Sessions

When you build web applications, you often fetch data from users using forms on the website. Flask's object request can handle that data. `request.form` can be used to access fields of the form.

You might want to store some custom information about a user in his browser. Cookies will help you do that. But to use cookies plainly can be a security risk. Hence it is recommended to use Sessions. Sessions allow you to store information specific to a user from one request to the next. The advantage of sessions is that it signs the cookie cryptographically so that they cannot be copied or stolen. To encrypt a session, you need to assign an Application secret. This must be a random string stored on your computer and should be stored securely as it can be used to decrypt all the sessions.

```
from flask import request

@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                        request.form['password']):
            return log_the_user_in(request.form['username'])
        else:
            error = 'Invalid username/password'
    # the code below is executed if the request method
    # was GET or the credentials were invalid
    return render_template('login.html', error=error)
```

## CHAPTER – 5 (Jinjer Templating)

### Jinjer templating

Flask comes packaged with the powerful Jinja templating language. Jinja2 is a modern day templating language for Python developers. It was made after Django's template. It is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Templating language essentially contain variables as well as some programming logic, which when evaluated (or rendered into HTML) are replaced with actual values. The variables and/or logic are placed between tags or delimiters. For example, Jinja templates use `{% ... %}` for expressions or logic (like for loops), while `{{ ... }}` is used for outputting the results of an expression or a variable to the end user. The latter tag, when rendered, is replaced with a value or values, and is seen by the end user.

### What are Templates

Back in the days, servers used to have a collection of files, like HTML files, which were sent over as requested by clients. These were static data being sent over.

Now, in the modern web world, we have less of static data and more of dynamic data being requested from clients and therefore sent by the server. The web totally depends on what the client is asking for, and on which user is signing in and who is logging out. So, Jinja2 templating is being used.

A template contains variables which are replaced by the values which are passed in when the template is rendered. Variables are helpful with the dynamic data.

The structure of your application helps to keep your code organised and accessible. Flask expects the templates directory to be in the same folder as the module in which it was created.

### Delimiters

`{%....%}` are for statements

`{{....}}` are expressions used to print to template output

`{#....#}` are for comments which are not included in the template output

`#....##` are used as line statements

Location : app/templates/index.html

```
<!DOCTYPE html>
<html>
<p>Welcome {{ username }}</p>
</body>
</html>
```

This is a very simple HTML file. Variables are like the placeholders which store dynamic data or values. These values may be passed or provided as arguments in render\_template() call. Rendering is a process of filling these placeholders with actual data. As HTML file is rendered, data is sent to a client.

Location : app/routes.py

```
from flask import Flask, render_template
app = Flask(__name__, template_folder='')
@app.route('/user/<username>')
def index(username):
    return render_template('index.html', username=username)
if __name__ == '__main__':
    app.run(debug=True)
```

The syntax is: flask.render\_template(template\_name\_or\_list,\*\*content)

This renders a template from the templates folder with the given content.

template\_name\_or\_list is the name of the template to be rendered or iterated.

\*\* means it is looking for keyword arguments.

Values can also be passed to the templates via the content dictionary. To print the value of variable, we use a{{ username }}. To access the variable's attributes, we can either use {{ username.surname }} or {{ username['title'] }}.

### **Conditional and loop Statements**

So long we've seen only seen one of the many advantages of Jinja 2. To control the flow of the program, we create a structure which is controlled by conditional statements. {%....%} these are the placeholders from which all the conditions are executed.

```

<!DOCTYPE html>
<html>
<head>
  <title>Conditions</title>
</head>
<body>

<ul>
  {% for name in list_example %}
  <li>{{ name }}</li>
  {% endfor %}
</ul>

<ol>
  {% for name in list_example %}
  <li>{{ name }}</li>
  {% endfor %}
</ol>

</body>
</html>

```

So, for loop is used to iterate over the list\_example and print it with this placeholder{{ name }}.

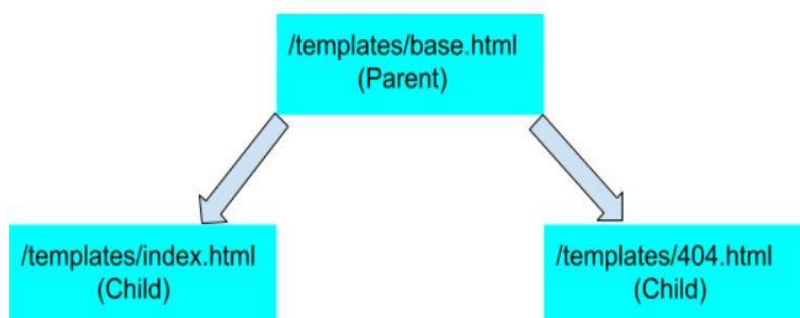
To end for loop use {%endfor%} and to end if loop, use {%endif%}.

### Template Inheritance

Jinja 2 supports Template Inheritance, which is one of the most powerful and useful features of any template engine. It means one template can inherit from another template.

Nowadays, websites require the same layout for different pages. Almost every website has a navigation bar attached to its page. To not repeat the code, we use inheritance feature because it saves us a lot of time and also reduces work.

A base template contains the basic layout which is common to all the other templates, and it is from this base template we extend or derive the layout for other pages.



Structure of Template Inheritance

```

<!-- "Parent Template" -->
<!DOCTYPE html>
<html>
<head>
  {% block head %}
  <title>{% block title %}{% endblock %}</title>
  {% endblock %}
</head>
<body>
  {% block body %}{% endblock %}
</body>
</html>

```

Template Inheritance uses `{% block %}` tag to tell the template engine to override the common elements of your site via child templates. `base.html` is the parent template which is the basic layout and on which you can modify using the child templates, which is used to fill empty blocks with content. `base.html` is a general layout of the Web page.

```

<!-- "Child Template" -->
{% extends "base.html" %}
{% block title %} Index {% endblock %}
{% block head %}
  {{ super() }}
{% endblock %}
{% block body %}
  <h1>Hello World</h1>
  <p>Welcome to my site.</p>
{% endblock %}

```

The `{% extend %}` must be the first tag in the child templates. This tag tells the template engine that this template extends from the parent template or ( `base.html` ). `{% extend %}` represents the inheritance characteristic of Jinja 2. So now the `{% extend "base.html"%}` first searches for the template mentioned and then the child template `index.html` overrides it with a different data.

```
{% extends 'base.html' %}

{% block title %} Page Not Found {% endblock %}

{% block body %}
  <h1>404 Error :(</h1>
  <p>What you were looking for is just not there.<p>
  <a href="{{ url_for('index') }}">go somewhere nice</a>
{% endblock %}
```

## **CHAPTER – 6 (JSON AND REST CALL)**

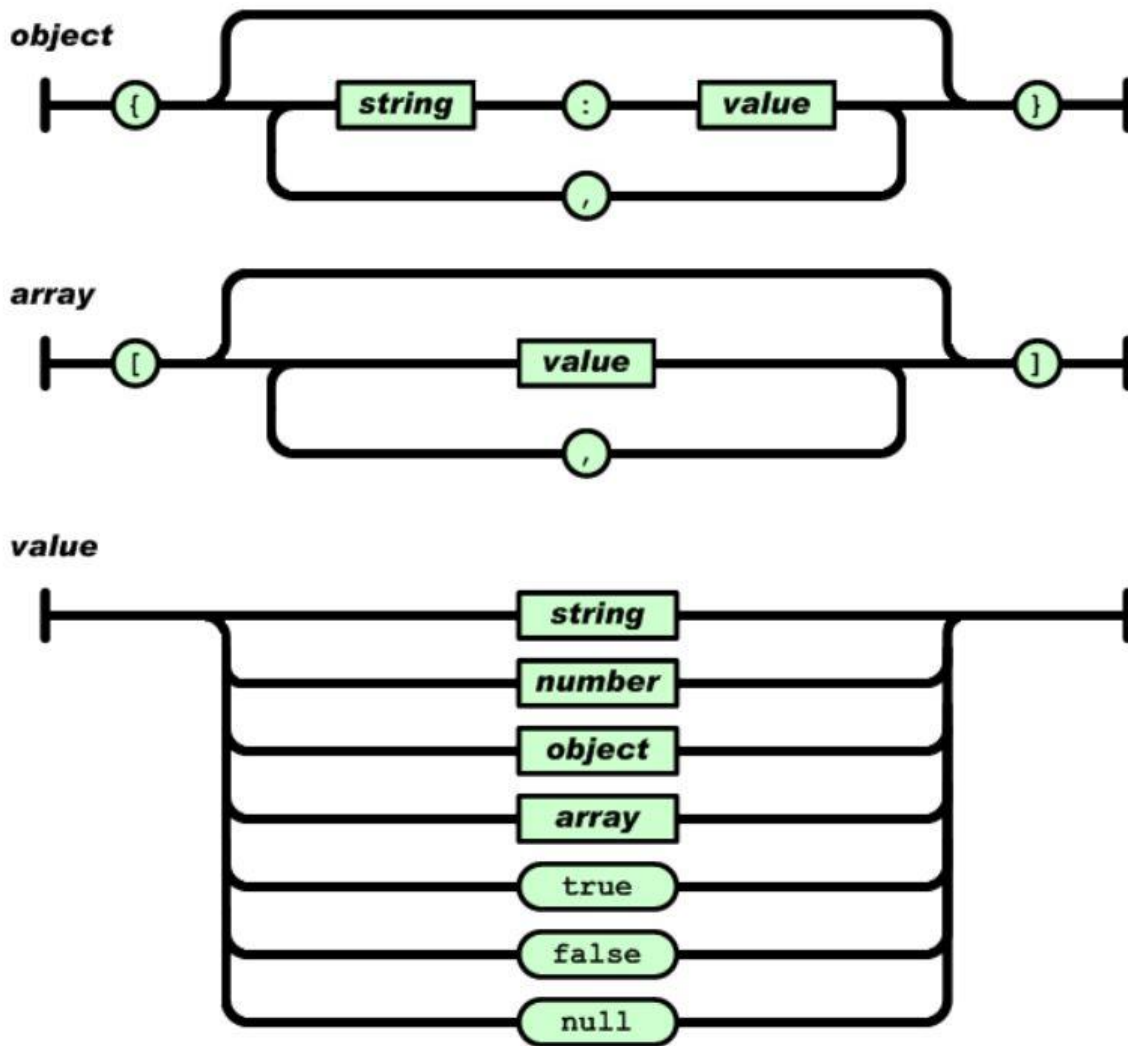
### **JSON**

JSON stands for JavaScript Object Notation. It is a lightweight format for storing and transporting data. It is often used when data is sent from a server to a web page. It is "self-describing" and easy to understand. It is based on a subset of JavaScript language (the way objects are built in JavaScript) it was designed for human-readable data interchange. It has been extended from the JavaScript scripting language. The filename extension is .json. JSON Internet Media type is application/json.

### **JSON Syntax Rules**

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays





JSON syntax example:

```
{ "book": [  
  {  
    "id": "01",  
    "language": "Java",  
    "edition": "third",  
    "author": "Herbert Schildt"  
  },  
  {  
    "id": "07",  
    "language": "C++",  
  }  
]
```

```
"edition": "second",  
"author": "E.Balagurusamy"  
}  
]  
}
```

### **JSON is built on two structures:**

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

### **Why use JSON?**

- Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.
- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.
- JavaScript has a built in function to convert a string, written in JSON format, into native JavaScript objects . So, if data is received from a server, in JSON format it can be used like any other JavaScript object
- XML is much more difficult to parse than JSON.

### **Where JSON is used?**

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

### **REST CALL**

REpresentational State Transfer

REST, or REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server

### **Separation of Client and Server**

In the REST architectural style, the implementation of the client and the implementation of the server can be done independently without each knowing about the other. This means that the code on the client side can be changed at any time without affecting the operation of the server, and the code on the server side can be changed without affecting the operation of the client.

As long as each side knows what format of messages to send to the other, they can be kept modular and separate. Separating the user interface concerns from the data storage concerns, we improve the flexibility of the interface across platforms and improve scalability by simplifying the server components. Additionally, the separation allows each component the ability to evolve independently.

By using a REST interface, different clients hit the same REST endpoints, perform the same actions, and receive the same responses.

### **Statelessness**

Systems that follow the REST paradigm are stateless, meaning that the server does not need to know anything about what state the client is in and vice versa. In this way, both the server and the client can understand any message received, even without seeing previous messages. This constraint of statelessness is enforced through the use of resources, rather than commands. Resources are the nouns of the Web - they describe any object, document, or thing that you may need to store or send to other services.

Because REST systems interact through standard operations on resources, they do not rely on the implementation of interfaces.

These constraints help RESTful applications achieve reliability, quick performance, and scalability, as components that can be managed, updated, and reused without affecting the system as a whole, even during operation of the system.

REST requires that a client make a request to the server in order to retrieve or modify data on the server.

### **Guiding Principles of REST**

**Client-server** – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

**Stateless** – Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.

**Cacheable** – Cache constraints require that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.

**Uniform interface** – By applying the software engineering principle of generality to the component interface, the overall system architecture is simplified and the visibility of interactions is improved. In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behavior of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

**Layered system** – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting.

**Code on demand (optional)** – REST allows client functionality to be extended by downloading and executing code in the form of applets or scripts. This simplifies clients by reducing the number of features required to be pre-implemented

## CHAPTER- 7

### STREAMLIT

Streamlit is an open-source Python library that makes it easy to build beautiful custom web-apps for machine learning and data science. Streamlit watches for changes on each save and updates the app live while you're coding. Code runs from top to bottom, always from a clean state, and with no need for callbacks. It's a simple and powerful app model that lets you build rich UIs incredibly quickly.

#### **Awesome Development flow of streamlit**

Every time you want to update your app, just save the source file. When you do that, Streamlit detects if there is a change and asks you whether you want to rerun your app. Choose “Always rerun” at the top-right of your screen to automatically update your app every time you change its source code. This allows you to work in a fast interactive loop: type some code, save it, try it out live, then type some more code, save it, try it out, and so on until satisfied with the results. This tight loop between coding and viewing results live is one of the ways Streamlit is easy to use.

#### **Caching in Streamlit**

The Streamlit cache allows your app to execute quickly even when loading data from the web, manipulating large datasets, or performing expensive computations. To use the cache, just wrap functions in the `@st.cache` decorator. When a function is marked with the `@st.cache` decorator, it tells Streamlit that whenever the function is called it needs to check a few things:

- The input parameters that you called the function with
- The value of any external variable used in the function
- The body of the function.
- The body of any function used inside the cached function.

#### **App model of Streamlit**

- Streamlit apps are Python scripts that run from top to bottom
- Every time a user opens a browser tab pointing to your app, the script is re-executed
- As the script executes, Streamlit draws its output live in a browser
- Scripts use the Streamlit cache to avoid recomputing expensive functions, so updates happen very fast

- Every time a user interacts with a widget, your script is re-executed and the output value of that widget is set to the new value during that run.

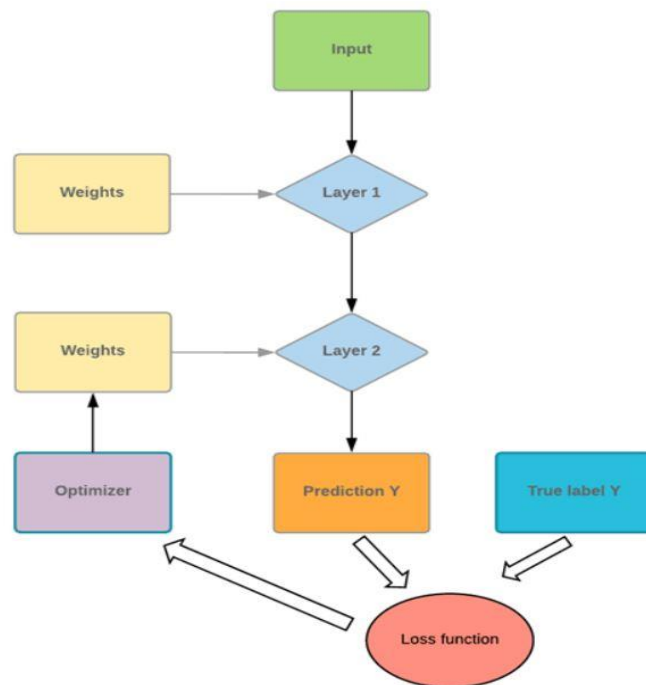
## CHAPTER – 8 (Nueral Network)

### What is Artificial Neural Network?

An Artificial Neural Network(ANN) is composed of four principal objects:

1. Layers: All the learning occurs in the layers. There are 3 layers
  - 1) Input
  - 2) Hidden
  - 3) Output
2. Feature and label: Input data to the network(features) and output from the network (labels)
3. Loss function: Metric used to estimate the performance of the learning phase
4. Optimizer: Improve the learning by updating the knowledge in the network

A neural network will take the input data and push them into an ensemble of layers. The network needs to evaluate its performance with a loss function. The loss function gives to the network an idea of the path it needs to take before it masters the knowledge. The network needs to improve its knowledge with the help of an optimizer.



## Loss function

After you have defined the hidden layers and the activation function, you need to specify the loss function and the optimizer.

For binary classification, it is common practice to use a binary cross entropy loss function. In the linear regression, you use the mean square error.

The loss function is an important metric to estimate the performance of the optimizer. During the training, this metric will be minimized. You need to select this quantity carefully depending on the type of problem you are dealing with.

## Optimizer

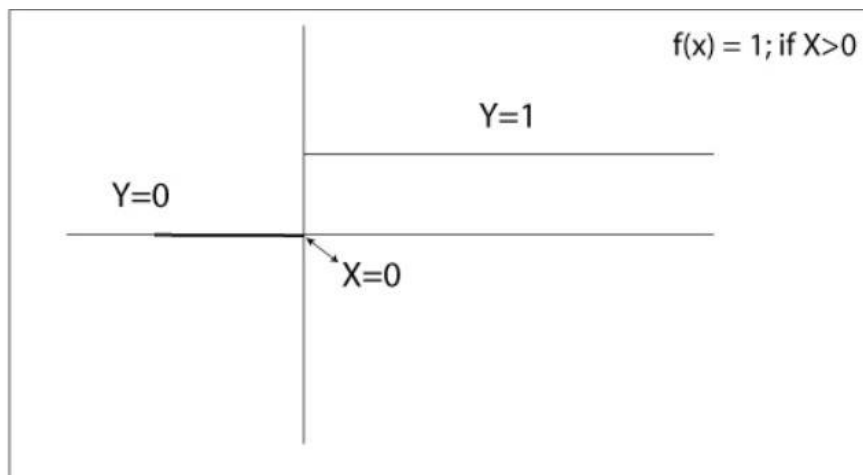
The loss function is a measure of the model's performance. The optimizer will help improve the weights of the network in order to decrease the loss. There are different optimizers available, but the most common one is the Stochastic Gradient Descent and Adam.

## Activation function

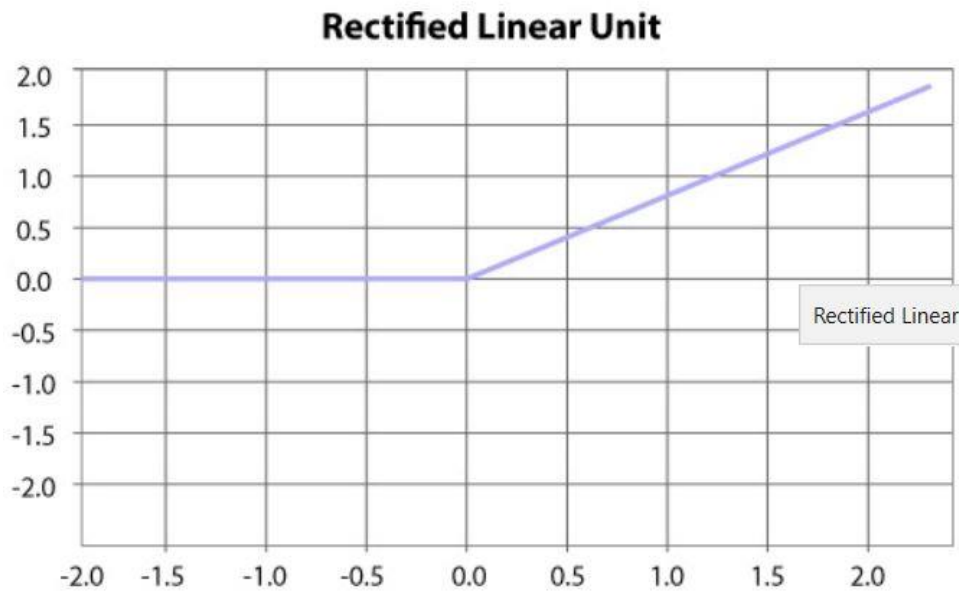
The activation function of a node defines the output given a set of inputs. You need an activation function to allow the network to learn non-linear pattern. A common activation function is a Relu, Rectified linear unit. The function gives a zero for all negative values.

Some of activation functions are :-

1. Binary Step Activation Function: This activation function is very basic and it comes to mind every time if we try to bound output. It is basically a threshold based classifier, in this, we decide some threshold value to decide output that neuron should be activated or deactivated. Below we decide the threshold value to 0.

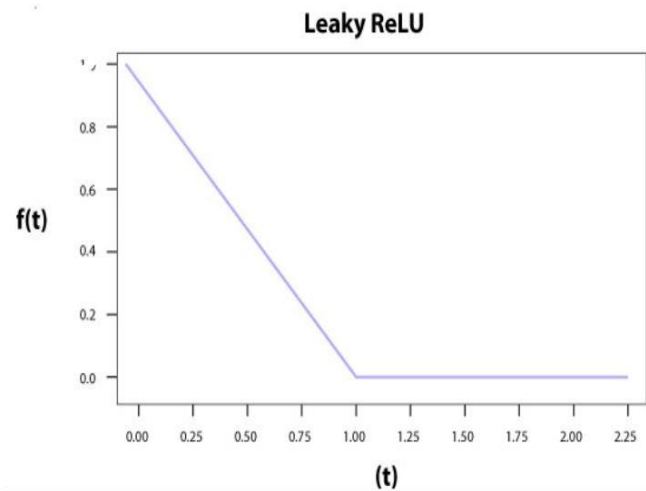


2. Rectified linear unit Function: Rectified linear unit or ReLU is most widely used activation function right now which ranges from 0 to infinity, All the negative values are converted into zero, and this conversion rate is so fast that neither it can map nor fit into data properly which creates a problem. We use Leaky ReLU function instead of ReLU to avoid this unfitting, in Leaky ReLU range is expanded which enhances the performance.

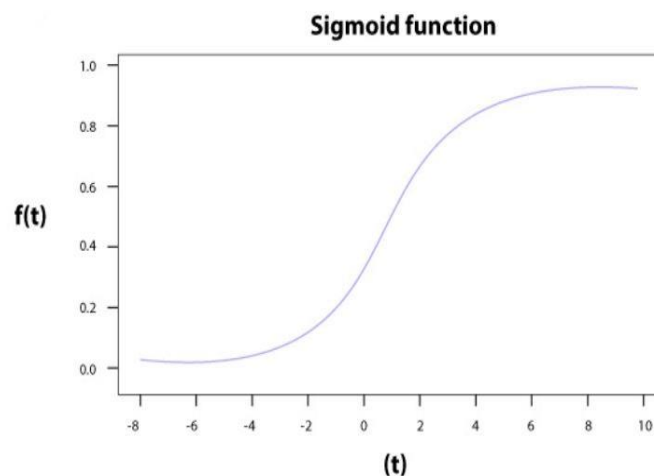


3. Leaky ReLU: We needed the Leaky ReLU activation function to solve the 'Dying ReLU' problem, as discussed in ReLU, we observe that all the negative input values turn into zero very quickly and in the case of Leaky ReLU we do not make all negative inputs to zero but to a value near to zero which solves the major issue of ReLU activation function.

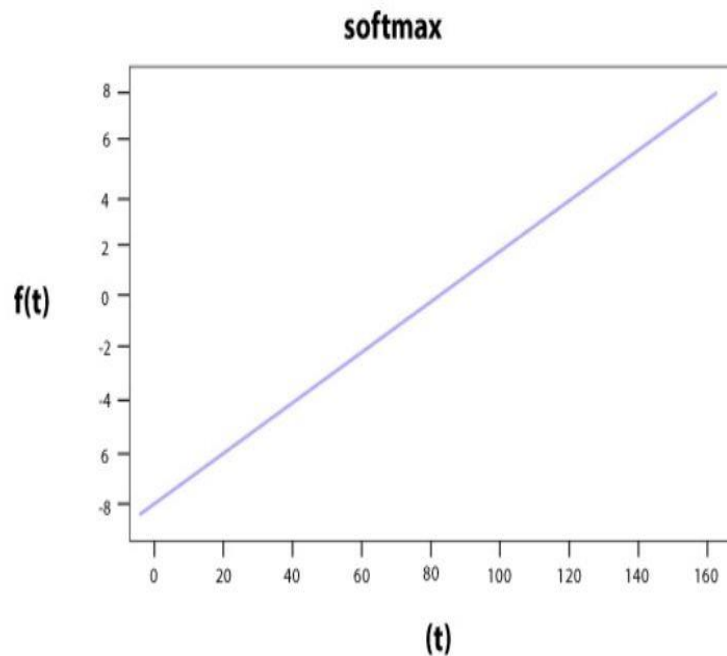




4. Sigmoid Function: The sigmoid activation function is used mostly as it does its task with great efficiency, it basically is a probabilistic approach towards decision making and ranges in between 0 to 1, so when we have to make a decision or to predict an output we use this activation function because of the range is the minimum, therefore, prediction would be more accurate. The equation for the sigmoid function is  $f(x) = 1/(1+e^{-x})$ . The sigmoid function causes a problem mainly termed as vanishing gradient problem which occurs because we convert large input in between the range of 0 to 1 and therefore their derivatives become much smaller which does not give satisfactory output. To solve this problem another activation function such as ReLU is used where we do not have a small derivative problem.



5. Softmax is used mainly at the last layer i.e output layer for decision making the same as sigmoid activation works, the softmax basically gives value to the input variable according to their weight and the sum of these weights is eventually one. For Binary classification, both sigmoid, as well as softmax, are equally approachable but in case of multi-class classification problem we generally use softmax and cross-entropy along with it.



## **Limitation of Neural network**

### **Overfitting**

A common problem with the complex neural net is the difficulties in generalizing unseen data. A neural network with lots of weights can identify specific details in the train set very well but often leads to overfitting. If the data are unbalanced within groups (i.e., not enough data available in some groups), the network will learn very well during the training but will not have the ability to generalize such pattern to never-seen-before data.

There is a trade-off in machine learning between optimization and generalization.

Optimize a model requires to find the best parameters that minimize the loss of the training set.

Generalization, however, tells how the model behaves for unseen data.

To prevent the model from capturing specific details or unwanted patterns of the training data, you can use different techniques. The best method is to have a balanced dataset with sufficient amount of data. The art of reducing overfitting is called regularization.

### **Network size**

A neural network with too many layers and hidden units are known to be highly sophisticated. A straightforward way to reduce the complexity of the model is to reduce its size. There is no best practice to define the number of layers. You need to start with a small amount of layer and increases its size until you find the model overfit.

### **Weight Regularization**

A standard technique to prevent overfitting is to add constraints to the weights of the network. The constraint forces the size of the network to take only small values. The constraint is added to the loss function of the error. There are two kinds of regularization:

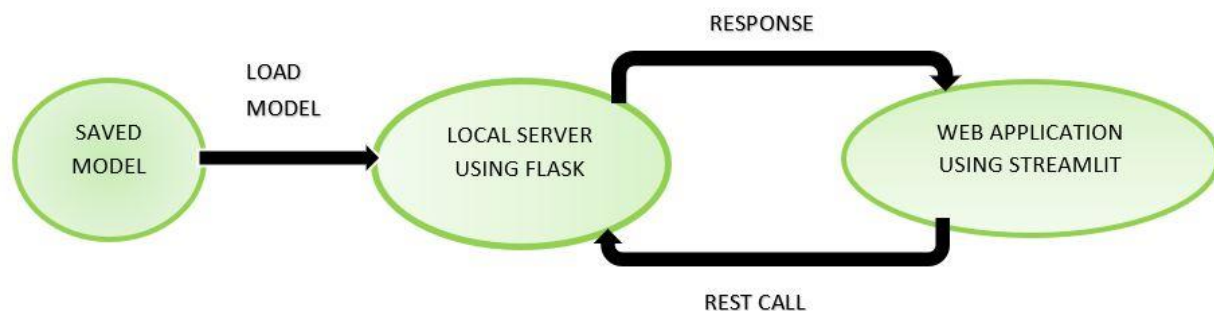
### **Dropout**

Dropout is an odd but useful technique. A network with dropout means that some weights will be randomly set to zero. Imagine an array of weights [0.1, 1.7, 0.7, -0.9]. If the neural network has a dropout, it will become [0.1, 0, 0, -0.9] with randomly distributed 0. The parameter that controls the dropout is the dropout rate. The rate defines how many weights to be set to zeroes. Having a rate between 0.2 and 0.5 is common.

## CHAPTER – 9 (CODE SNIPPETS AND EXPLANATION)

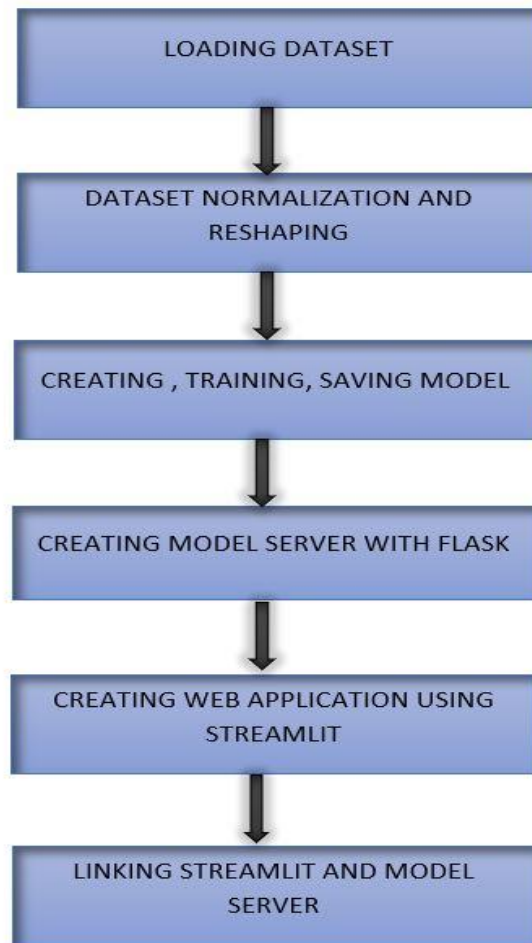
### HOW OUR CODE WORKS

We build and train our model to recognize the handwritten digits. We write code to train the model and Save model as model.h5 ,So that we can use it afterwards. We create server.py file using flask and this file acts as a server and Then we wrtie code for app.py which is used for web application. This web application and server communicates with each other using the rest call. Web application requests for the model and server responds to it.



The Neural network with 1 input layer , 2 hidden layer, 1 ouptut layer will be created .We will visulaize the output values of all these layers of all the nodes.Every node value will be represented with a grayscale color depicting its value. We will create server with the help of flask to serve our Neural network Model and web application with the help of Streamlit and to obtain output of hidden layers we will use keras functional API.We will train our model with the help of keras. Dataset that we will use is EMNIST dataset, it is a dataset of handwritted digits of size 28X28. We will get a input image from our dataset and then we will try to predict it by clicking on the prediction button. And the output will be visual representation of all the layers in our model and also the output layer, which will be shown tells what is our output. It basically helps us to visualize our neural network.We can also view our inputted image from sidebar.

The main tasks we perform to build this application is as follows in the form of flowchart:-



## CODE:

### CODE FOR TRAINING MODEL

```
import tensorflow as tf
import keras
import matplotlib.pyplot as plt
import numpy as np
#download data
(x_train,y_train),(x_test,y_test)=tf.keras.datasets.mnist.load_data()
plt.figure(figsize=(10,10))
for i in range(0,16):
    plt.subplot(4,4,i+1)
    plt.imshow(x_train[i],cmap='binary')
    plt.xlabel(str(y_train[i]))
```

```

plt.xticks([])
plt.yticks([])
plt.show()
#normalize data
x_train=np.reshape(x_train,(60000,28*28))
x_test=np.reshape(x_test,(10000,28*28))
x_train=x_train/255
x_test=x_test/255
x_train.shape

#create nueral network

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(32,activation='sigmoid',input_shape=(784,)),
    tf.keras.layers.Dense(32,activation='sigmoid'),
    tf.keras.layers.Dense(10,activation='softmax')
])

model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

#train the model
_=model.fit(
    x_train,y_train,
    validation_data=(x_test,y_test),
    epochs=20,
    batch_size=1024,
    verbose=2,

```

```
)  
#save model  
model.save('model.h5')
```

**output**  
model.h5

## CODE FOR SERVER

```
import json  
import tensorflow as tf  
import numpy as np  
import random  
from flask import Flask,request  
  
app = Flask(__name__)  
  
model=tf.keras.models.load_model('model.h5')  
feature_model= tf.keras.models.Model(  
    model.inputs,  
    [layer.output for layer in model.layers]  
)  
_,(x_test,_) =tf.keras.datasets.mnist.load_data()  
x_test=x_test/255  
  
def get_prediction():  
    index=np.random.choice(x_test.shape[0])  
    image=x_test[index,:,:]  
    image_arr=np.reshape(image,(1,784))  
    return feature_model.predict(image_arr),image  
  
@app.route('/',methods=['GET','POST'])
```

```
def index():
    if request.method == 'POST':
        preds,image=get_prediction()
        final_preds=[p.tolist() for p in preds]
        return json.dumps({
            'prediction':final_preds,
            'image':image.tolist()
        })
if __name__=='__main__':
    app.run(debug=True)
```

## MODEL SERVER EXPLANATION

This is our model server

Modules which are being imported in this File are as follows:-

- json
- tensorflow
- numpy,
- random
- Flask
- Request

Trained model have been loaded from model.h5 file in model variable.

Now if we directly used model then we will get only the output values of output layer but as we have to visualize the whole model therefore we need output values of all the dense layers.

If we run model.predict() right now we have only the ouptut of final layer we do not have the output of all the layers yet, it will only give us class that is predicted but other than that we don't really know about other layers output.To solve this we will use Keras functional API

### What is keras Functional API

The Keras functional API is a way to create models that is more flexible than the tf.keras.Sequential API. The functional API can handle models with non-linear topology, models with shared layers, and models with multiple inputs or outputs.The main idea that a deep learning model is usually a directed acyclic graph (DAG) of layers. So the functional API is a way to build graphs of layers.



We are using functional API so that we can have model with multiple outputs We will create another model let it call as feature\_model it will take the same input but it will give us output of all the layers We will use Model class from keras.models

### **Model class**

#### **Arguments**

inputs: The input(s) of the model: a keras.Input object or list of keras.Input objects.

outputs: The output(s) of the model.

name: String, the name of the model

We have to use keras functional API which will have same input as keras input model but it will have output of all the layers apart from input layer. First dense layer , second dense layer third dense layer. First argument is input and second argument is output, we have layers in model.layers we will iterate over them. Output will list of layer outputs for all the layers

We are running model server and web application is running separately. They will interact via a rest call to model server. We will download dataset again but this time only the test examples

Then we normalize it we have not reshaped until now.

#### **Get prediction function:**

To predict image we will randomly select it from our dataset. Then we will reshape this image.

After that we will return the predicted image and the input image itself. Now if request method is POST then what will we do we will get our prediction. We are converting numpy arrays to list so that we can transfer data using Json.

#### **json.dumps()**

json.dumps() function converts a Python object into a json string. The full-form of JSON is JavaScript Object Notation. It means that a script (executable) file which is made of text in a programming language, is used to store and transfer the data. Python supports JSON through a built-in package called json. To use this feature, we import the json package in Python script. JSON syntax is basically considered as a subset of JavaScript syntax; it includes the following – Data is represented in name/value pairs. Curly braces hold objects and each name is followed by ':' (colon), the name/value pairs are separated by , (comma). Square brackets hold arrays and values are separated by , (comma).

We have made debug = true so that we can debug easily Our app.py is making a REST call to server.py.

## CODE FOR WEB APPLICATION

```
import streamlit as st
import json
import requests
import matplotlib.pyplot as plt
import numpy as np

URI='http://127.0.0.1:5000'

st.title("Neural Network Visualizer")
st.sidebar.markdown('## Input Image')

if st.button('Get random prediction'):
    response=requests.post(URI,data={ })
    response=json.loads(response.text)
    preds=response.get('prediction')
    image=response.get('image')
    image=np.reshape(image,(28,28))

    st.sidebar.image(image,width=150)
    for layer,p in enumerate(preds):
        numbers=np.squeeze(np.array(p))
        plt.figure(figsize=(32,4))
        if(layer==2):
            row=1
            col=10
        else:
            row=2
            col=16
```

```

for i,number in enumerate(numbers):
    plt.subplot(row,col,i+1)
    plt.imshow(number * np.ones([8, 8, 3]).astype('float32'))
    plt.xticks([])
    plt.yticks([])
    if layer==2:
        plt.xlabel(str(i),fontsize=40)
plt.subplots_adjust(wspace=0.05,hspace=0.05)
plt.tight_layout()
st.text('Layer {}'.format(layer+1))
st.pyplot()

```

## WEB APPLICATION EXPLANATION

aap.py

Following modules are imported in this file :

- streamlit
- Json
- Requests
- Matplotlib
- Numpy

As soon as you run the script as shown above, a local Streamlit server will spin up and our app will open in a new tab your default web browser. The app is your canvas, where you'll draw charts, text, widgets, tables, and more.

### How streamlit is working

Streamlit apps are Python scripts that run from top to bottom Every time a user opens a browser tab pointing to your app, the script is re-executed As the script executes, Streamlit draws its output live in a browser Scripts use the Streamlit cache to avoid recomputing expensive functions, so updates happen very fast Every time a user interacts with a widget, your script is re-executed and the output value of that widget is set to the new value during that run.

First we add title “Neural Network Visualizer”. Then we add markdown “InputImage” in the sidebar

### **json.load()**

json.load() takes a file object and returns the json object. A JSON object contains data in the form of key/value pair. The keys are strings and the values are the JSON types. Keys and values are separated by a colon. Each entry (key/value pair) is separated by a comma. json.load(file\_object) It takes file object as a parameter and return json object. This json object can be thought of as a dictionary.

### **Requests.post()**

The post() method sends a POST request to the specified url. The post() method is used when you want to send some data to the server. We are making our post request to our ml server. URI specified is of ml server. As there is only one post request therefore it will return that. In response text we convert the response to json object. We take image from response and prediction from response. We will reshape our image as we have converted it to list so now we reshape it to 28\*28. We will show the image using streamlit. We will put this image on sidebar.

### **numpy.squeeze()**

numpy.squeeze() function is used when we want to remove single-dimensional entries from the shape of an array. numpy.squeeze(arr) parameter is input array and it returns squeezed [ndarray] The input array, but with all or a subset of the dimensions of length 1 removed. This is always a itself or a view into arr.

### **figure()**

The figure() function in pyplot module of matplotlib library is used to create a new figure. The parameter are the width, height in inches. This returns the Figure instance returned will also be passed to new\_figure\_manager in the backends.

### **numpy.astype()**

In order to change the dtype of the given array object, we will use numpy.astype() function. The function takes an argument which is the target data type. The function supports all the generic types and built-in types of data.

### **numpy.array()**

Using numpy.array() This function of the numpy library takes a list as an argument and returns an array that contains all the elements of the list.

## **Enumerate**

A lot of times when dealing with iterators, we also get a need to keep a count of iterations. Python eases the programmers' task by providing a built-in function `enumerate()` for this task.

It takes two parameters: `iterable` - a sequence, an iterator, or objects that supports iteration  
`start` (optional) - `enumerate()` starts counting from this number. If `start` is omitted, 0 is taken as `start`. `enumerate()` method adds counter to an iterable and returns it. The returned object is a `enumerate` object. we can convert `enumerate` objects to list and tuple using `list()` and `tuple()` method respectively.

## **numpy.ones**

`numpy.ones(shape, dtype = None)` : Return a new array of given shape and type, with ones.

Parameters are `shape` : integer or sequence of integers `dtype` : [optional, float (by default)] Data type of returned array and return type is `ndarray` of ones having given shape and datatype.

## **Subplot()**

The `subplots()` function in `pyplot` module of `matplotlib` library is used to create a figure and a set of subplots. Parameters are `rows`, `ncols` : These parameter are the number of rows/columns of the subplot grid. `Index`: it tells the index position. index starts at 1 in the upper left corner and increases to the right. This method return the following fig : This method return the figure layout. `ax` : This method return the axes. `Axes` object or array of `Axes` objects.

## **Imshow()**

The `imshow()` function in `pyplot` module of `matplotlib` library is used to display data as an image; i.e. on a 2D regular raster. This returns the `AxesImage`

## **xlabel , xtick , ylable , ytick**

Ticks are the markers denoting data points on axis.

The `xticks()` and `yticks()` function takes a list object as argument. The elements in the list denote the positions on corresponding axis where ticks will be displayed. Similarly, labels corresponding to tick marks can be set by `set_xlabel()` and `set_ylabel()` functions respectively.

## **CONCLUSION**

This chapter provides me an opportunity to do self-introspection of what value I have added to my knowledge and skill set and to the project.

## **CONCLUSION**

While doing this project I have realized it has helped me to learn how to intergrate technologies. Moreover I have learnt how to make web apps using python's web framework. I had understand that how frameworks makes your task easy once your learn it . Also I have learned to deploy machine learning model on the website. It has made me realized that computer science is interdisciplinary field. It has also make me realized that I can learn a lot using project based learning . It has provided me a expanded view from traditional way of web development using html,css,javascript and had motivated me to learn more new technologies . I have also learned how the rest call works . in terms of technologies I have learned html ,css ,javascript ,flask ,streamlit ,pandas , matplotlib, many concepts of deep learning etc. Exposing myself to various different libraries and framework has proved beneficial for me and I will try to learn more in the future. I have certainly improved my programming skills through this project. The code could be more documented , some of the problems posed by the project were a good challenge to solve.

## **FUTURE SCOPE**

There is a scope of adding fucntionality from which we can visualize nueral network diagram. Now we are inputting a random inage from dataset but we can add a functionality by which we can input our own image. We can aslo use this for visualiztion of other models besides the model which recognize digits. We can also expand by adding the same functionality for convolutional nueral networks. We can add more webpages to make the Site more engaging. We can host it on live server using the services such as heroku. While doing this project I have realized it has helped me to learn how to intergrate technologies. Moreover I have learnt how to make web apps using python's web framework. I had understand that how frameworks makes your task easy once your learn it . Also I have learned to deploy machine learning model on the website. It has made me realized that computer science is interdisciplinary field. It has also make me realized that I can learn a lot using project based learning . It has provided me a expanded view from traditional way of web development using html,css,javascript and had motivated me to learn more new technologies

## **BIBLOGRAPHY**

<https://dev.to/gajesh/the-complete-flask-beginner-tutorial-124i>

<https://codeburst.io/jinja-2-explained-in-5-minutes-88548486834e>

<https://towardsdatascience.com/streamlit-101-an-in-depth-introduction-fc8aad9492f2>

<https://www.codecademy.com/articles/what-is-rest>

<https://www.guru99.com/artificial-neural-network-tutorial.html>