Graphical User Interface for Tensorflow

Mentors: Monjoy Saha and Pooya Mobadersany

Student Name: Vikas Gola

Student Email: vikasgola2015@gmail.com

Student University: Indian Institute of Technology, Jammu

Organization: Emory University School of Medicine

Abstract

Tensorflow is one of the most famous open source deep learning library built by Google. It has been used by millions of deep learning experts. From the years, it has not been easy to make deep learning models and it is one of the major restriction for many people to implement new ideas in deep learning and check their results. This project takes this challenge of making deep learning models easily and tries to solve that.

The idea of this project is to make a software in which a user can make deep learning models in an easy way using a graphical user interface with backend supported by tensorflow. Through the graphical user interface, a user will able to add, delete, edit deep learning layers in a model. The main purpose of the project is to make the implementation of deep learning models quick and easy.

The software will be built using electron-js which is a framework for building cross-platform desktop apps with HTML, CSS, and JavaScript. It will have a drag-and-drop feature to build deep learning models in the form of a graph which will then converted to a python code by the software. The generated code will then be executed in the child process which trains the deep learning model and sends the metrics data(loss, accuracy) to the parent process which then plots the statistics.

Introduction

Deep learning has existed for more than 30 years but it is now at its peak. One of the biggest reason for the growth in Deep Learning is a decrease in the cost of processing power. Tensorflow, as a deep learning library, is used extensively all over the globe. It is has been developed and maintained by Google since 2015. It almost has all the required functions which a deep learning engineer or researcher needs. But still, it has been difficult and time consuming for beginners to write code in python on tensorflow. This project idea is to solve this by creating a graphical user interface for tensorflow. The details about the steps involved while building the project are described in the next sections. Current work status of this project is also mentioned in the next sections.

Literature Review

There is a number of libraries and tools for deep learning models e.g. Tensorflow, Theano, Keras, Caffe, Microsoft Cognitive Toolkit, PyTorch, etc. Tensorflow is one of the most used and efficient libraries in these. Tensorflow has grown and maintained over the years by Google. A number of tools for creating the deep learning models have come in this duration. Some of them with their advantages and disadvantages are discussed below.

- **Fabrik**

  According to its GitHub repo "Fabrik is an online collaborative platform to build, visualize and train deep learning models via a simple drag-and-drop interface. It allows researchers to collectively develop and debug models using a web GUI that supports importing, editing and exporting networks to popular frameworks like Caffe, Keras, and TensorFlow".

  Advantages

  - Gives the generated deep learning model code in Caffe, Keras, and Tensorflow.
  - Drag and Drop GUI
  - Open source

- Documented

Disadvantages

- Difficult to set up
- Limited deep learning layers
- No data loading and training visualization option
- Can't train or test models in it

- **Tensorbuilder**

  According to its GitHub repo "A drag and drop GUI tool written in Qt to rapidly prototype TensorFlow computational graphs, and instantly share your models to the community."

  Advantages

  - Drag and Drop GUI
  - The model can be checked for errors
  - Instant sharing option to community server where others can fork and edit the models.
  - Open source

  Disadvantages

  - Only for prototyping
  - Can't train or test models in it
  - No data loading and training visualization option
  - No code editor to edit the python code of models
  - No proper documentation

- **Deep learning studio**

  According to its website "Deep Learning Studio is a deep learning platform for creating and deploying AI."

  Advantages

  - Drag and Drop GUI
  - Can be used in browser or desktop
  - The model can be trained on their cloud or on the user desktop

Disadvantages

- Require NVIDIA GPU with compute capability > 3.0 to run.
- Dataset can only be uploaded in CSV format
- A model can't be trained on user cloud.
- Does not provide custom data loading and data manipulation layers.
- Not an open source project
- Does not have the option to get generated python code for any deep learning model

Objectives

These are the following objectives listed below which I will complete in my GSoC period:

- To make the process of creating deep learning models easy
- To make software easily available to users
- To create bug-free software
- To make deep learning coding-free
- A user-friendly interface for deep learning creation
- A collection of examples

Proposed Methodology

The project is divided into 5 parts. These are the following parts, which will be developed over the GSoC period, and the details of each part are described below:

- User Authentication

  This part consists of a page by which the user can log in in his account in software if a user already has the account, otherwise, the user has to create a new account and then log in. To create a new account, a user has to provide a username, password, dob, and his/her full name.

- Projects Management

  A page which shows a list of all created projects by the logged user with details(e.g. last modified, the name of the project, short description of the project) and shows user information which was given by the user at the signup process except for password. The user has options to create a new project and open a project from the listed projects. The user can also edit his info at this page.

- Drag and Drop Model Creation (Main Page)

  This part is the main page of software where the user will create deep learning models. The page consists of three left sidebars (only one of them can be viewed at an instant), one right sidebar, one header, and one graph container. These are the following important definitions which will be used in the description of parts of the page which I named above:

  - Model Graph

    This is a visual representation of the deep learning model which consist of Layer Node, Data Node, Output Node, and Node Connectors. This is a typical graph data structure made of nodes and edges.

  - Node Connector

    This is a directed edge which connects two or more than two nodes in a model graph. This shows the virtual data flow in the direction of the edge in a model graph.

  - Layer Node

    This is a node which can represent a deep learning layer in the model. E.g. Conv2D, LSTM, GRU, Input, etc. This type of nodes can only be connected to any other layer nodes except the input layer node which can be connected to one data node and one layer node.

- Data Node

  This type of node represents all the type of data loading pre-built functions, data manipulation pre-built functions or data related python functions that have been created by the user which can be connected to any other data nodes or an input layer node. E.g. LoadImage, LoadCSV, LoadAudio, ImageAugmentation etc.

- Output Node

  This node is used to show the output of the model which can be viewed in the model training and evaluation page. This node may represent any file, charts related to metric, accuracy, and loss.

The description of each part of the page which I listed above, will be as follows:

- Graph Container

  The graph container is a box which will contain a visual representation of a user's deep learning model(i.e. model graph), which is nothing but the visual representation of a graph data structure, made of different nodes(Layer Node, Data Node, and Output Node) and node connectors(also known as an edge).

- Left Sidebars

  This part of the page consists of a group of three sidebars which can be tabbed. The first sidebar has the list of all layer nodes and can be known as **Layer Sidebar**. The second bar has the list of all data nodes and can be known as **Data Sidebar**. Similarly, the third sidebar has the list of all output nodes and can be known as **Output Sidebar**.

- Right Sidebar

  This includes a list of inputs for arguments that have to pass on to a selected node of the model graph.

- Header

  The header is the part in the page which has the menu for a project and has project details. This has options to save and exit the opened project, etc.

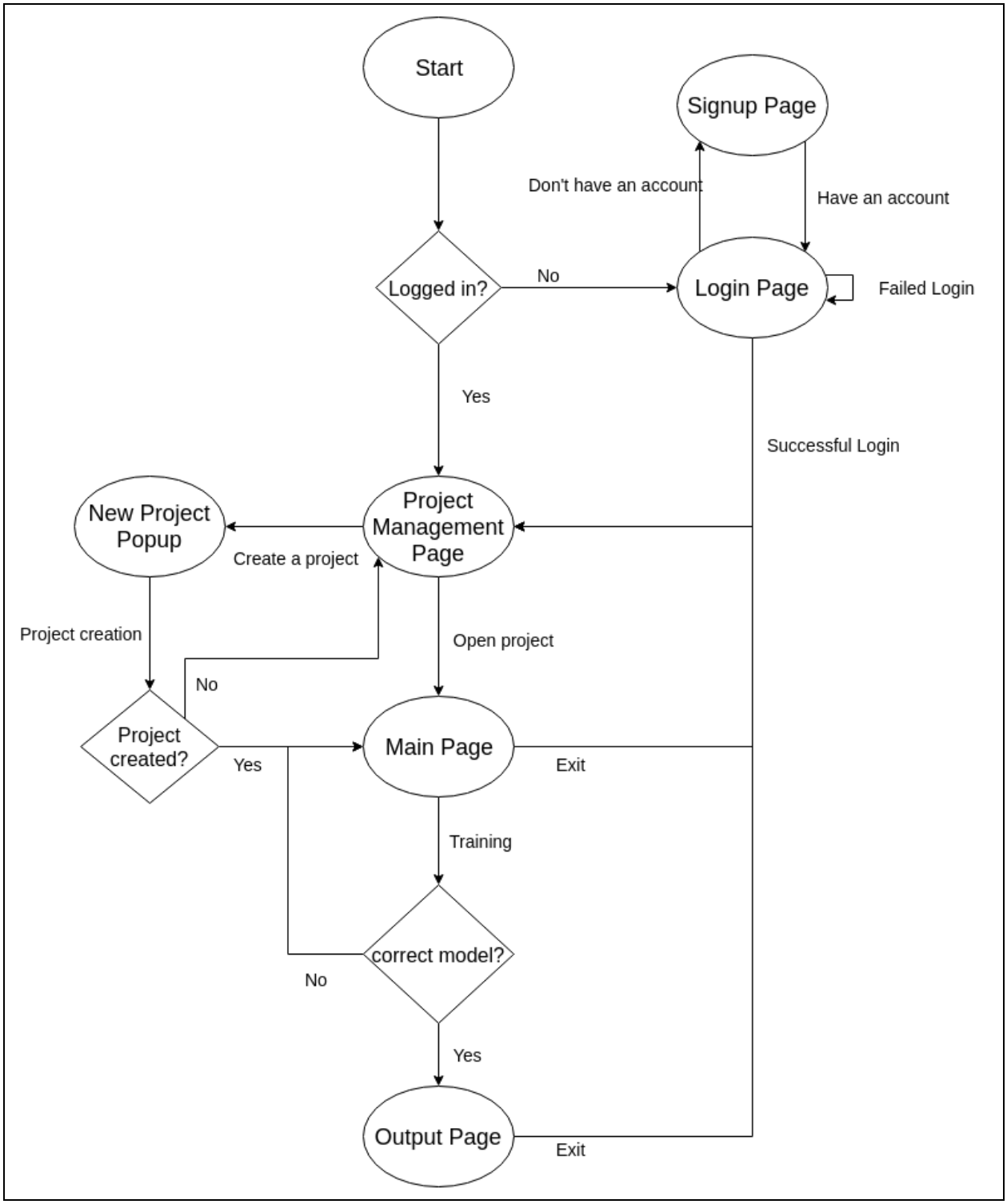
- Model Training and Evaluation (Output Page)

  This part has a page which shows the output of deep learning model after training. The output can consist of charts, final accuracy, loss, testing input results, etc. This page also provides options to load the model from the given path and save on a given path with trained weights.

- 10 Examples

  This part of the project is focused on developing examples for the final software. This part will help users to get started with software and understanding the robustness of software. The example can be on MNIST dataset, CIFAR10 dataset, IMDB sentiment classification task, etc.

The software will be built in ElectronJS (a cross-platform desktop app framework) which uses JavaScript, HTML, and CSS. The software will be made of 4 pages which are User Auth. Page, Project Management Page, Main Page, and Output Page.



The Graph container on the main page will be an HTML canvas on which HTML list from left sidebars will be dragged and will be dropped on to this canvas. On drop, a node using one of

node editor js library will be created. This node editor library feature of connecting nodes will be used to let users connect the nodes.

After completion of the model, the python code will be generated from the model graph. To create python code from the model graph, the graph would be traversed using graph traversing algorithm. There are three reasons to do this.

The first reason is to detect the cycles in the model graph as it cannot contain cycles. The DFS can be used to do this task and here is the algorithm for finding cycles in the graph.

```
// Algorithm for detecting cycles in the graph
DFS (G):
   for each vertex u belongs to G.V:
      u.color = WHITE
      u.pi = NIL
   time = 0
   for each vertex u belongs to G.V:
       if u.color == WHITE:
           DFS-VISIT(G, u)
DFS-VISIT(G, u):
   time = time + 1
   u.d = time
   u.color = GRAY
   for each v belongs to G. Adj[u]:
       if v. color == WHITE:
           v.pi = u
            DFS-VISIT(G, v)
      else:
           //cycle detected
   u.color = BLACK // blacken u;
   time = time + 1
   u.f = time
```

The second reason is to detect the disconnected graph i.e. there has to be at least one path from every node to every other node in the graph.

```
// Algorithm for detecting disconnected graphs
DFS(v):
  visited[v] = true
  for each successor node u in v:
    if visited[u] = false:
      DFS(u)

checkAllVisited():
  for each node v in graph:
    if visited[v] == false:
        return false
  return true

checkAllConnected():
   for each starting node y in graph:
     DFS(y)
   return checkAllVisited()
```

The third and most important reason is to generate the python code in the tensorflow for the model. Again, DFS can be used for this purpose in which traversal will be done from every input layer node.

```
DFS(v):
  visited[v] = true
  for each successor node u in v:
    if visited[u] = false:
      DFS(u)
```

Present Status of Work(Prototype)

This is the following work listed below which is already completed before April 9:

- User Authentication is completed.
- The frontend of the Project Management page is completed.
- The basic Main and output pages are also completed.

- Deep learning models can be created with basic Conv2d, Input, Output, Dense Layers. The model can be trained and basic loss, accuracy chart can be seen in Output Page.
- Simple deep learning model with two dense layers was trained to calculate the sum of two input integers.
- Deep learning model with Convolution, Pooling, Dropout, and Dense layers was trained on MNIST dataset.

The demo videos of the prototype can be seen here.

Timeframe

Before 27 May(before coding officially begins)

- Understanding the project idea
- Retrieve all the requirements of a user
- Discuss my idea with mentors and get feedback from them and update the idea accordingly.
- Create a flowchart of software, create a GitHub repo and start coding
- Create an authentication page and complete email authentication in it.
- Integrate google login in the authentication page.
- Create project management page and complete frontend.
- Add the functions to create configuration files of the project and load them.

27 May - 2 June (after coding officially begins)

- Improve created frontend of Main Page consists of Graph Container, Left Sidebars, Header, and Right Sidebar.
- Fill the Left Sidebars(Data Sidebar, Layer Sidebar, and Output Sidebar) in Main Page with the list of all nodes names that can be dragged and dropped.

3 June - 9 June

- Attach arguments input fields of nodes to Right Sidebar in respect of node.
- Add Node Connectors (or model graph edge) to Graph Container.

- Add option to delete nodes and node connectors from the model graph. Also, add the option to edit the arguments of all nodes.

10 June - 16 June

- Generate python code from layer nodes from the model graph.
- Test and debug the model python code generation from layer nodes.
- Get arguments value from Right Sidebar of the node and pass it on tensorflow model python code.

17 June - 23 June

- Write functions of Data Nodes that are related to the loading of any data (Image, Image Folder, Audios, CSVs, Text Files, Videos).
- Add Custom Data Node functionality in which a user can write his own python function. i.e. Integration of Code editor

24 June - 30 June

- **Phase 1 Evaluation**
- Write functions of Data Nodes that are related to data manipulation of Images and Videos.
- Test and debug the Data Nodes of data manipulation of Image and Videos.

1 July - 7 July

- Write functions of Data Nodes that are related to data manipulation fo Audios, CSVs and Text Files.
- Test and debug all the Data Nodes.

8 July - 14 July

- Add option to provide Model arguments to final tensorflow model. Examples of Model arguments are learning_rate, batch_size, optimizer and loss function.
- Combine python code generation of Data Nodes and Layer Nodes.

- Test and debug the combined python code generation of Data Nodes and Layer Nodes.

15 July - 21 July

- Add python child process creation on starting training of the model.
- Add output return functionality in real time from the child python model training process.
- Test and debug the python code generation and process running of the model.

22 July - 28 July

- **Phase 2 Evaluation**
- Write functions of Output Nodes.

29 July - 4 August

- Test and debug the Output Nodes on adding in the model graph.
- Test and debug the python code generation of the model graph after adding the output nodes.

5 August - 11 August

- Create Training and Evaluation Page and complete the frontend.
- Attach the output of the model graph to this page.
- Deep Learning Model weights saving and loading functionality.

12 August - 18 August

- Testing and debugging of Software.
- Writing examples.

19 August - 26 August (Final Week)

- More Testing and debugging of software.
- **Final Student Evaluation**

CV

I am Vikas Gola, a 3rd-year undergraduate student at Indian Institute of Technology, Jammu(India). I entered the world of open source last year and from then it has been a great learning period for me. It has helped me a lot to show my experience and skills in the world, and also to learn new skills. It would be a great summer to put my skills to this project. I have done a course in Artificial Intelligence in university and have done a number of assignments in Python with the tensorflow, and Keras.

Academic Qualifications

- 2016 - Present
  Degree: B.Tech in Computer Science and Engineering
  University: Indian Institute of Technology Jammu

Experience

- mentored 130 participants of Competitive_Coding project in GirlScript Summer of Code 2018 (website).
- Contributed to Anna-Assistant project in GirlScript Summer of Code 2018.
- Over **450 Github contributions** last year on GitHub in over **40 repositories**.
- Developed an Attendance App for Kilimanjaro Pvt. Ltd. (Delhi, India).
- Secured **1st Rank** in Coding Hackathon at Inter IIT Tech Meet 2018.
- Secured place in top 20 teams in Pan IIT Hackathon 2019.

References

- Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (June 1990). *Introduction to Algorithms*. MIT Press.
- *Deep Cognition*. (n.d.). Retrieved from https://deepcognition.ai/
- *Depth-first search*. (n.d). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Depth-first_search
- *Electron*. (n.d.). Retrieved from https://electronjs.org/
- *TensorFlow*. (n.d.). Retrieved from https://www.tensorflow.org/
- Deep learning. (n.d). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Deep_learning
- TensorBuilder. (n.d). Retrieved from GitHub: https://github.com/TommyX12/tensorbuilde
- Fabrik. (n.d). Retrieved from GitHub: https://github.com/Cloud-CV/Fabrik