

Technical Report of TensorFlow-GUI

Google Summer of Code 2019

Author: Vikas Gola

Mentors: Monjoy Saha (PhD), Pooya Mobadersany

Organization: Biomedical Informatics, Emory University

Introduction

TensorFlow-GUI is a desktop software to build the deep learning models easily and efficiently. It uses tensorflow for creating models in python. It has inbuilt support for many famous datasets which can be used easily for training models. It is built using [Electron](#) which is an open source framework. It also uses [CodeMirror](#) for text editor, [KonvaJS](#) for creating the node editor, and [tensorboard](#) to show the progress of tensorboard. In this document, we will look at how it can be used to build models and how it has been built.

How to setup and use!

A. Required libraries/ packages

- Python 3
- Natsort
- Tensorflow
- Pandas
- NodeJS
- Electron
- JQuery
- JQuery-ui-dist
- Rimraf
- Sweetalert

B. Supported OS details

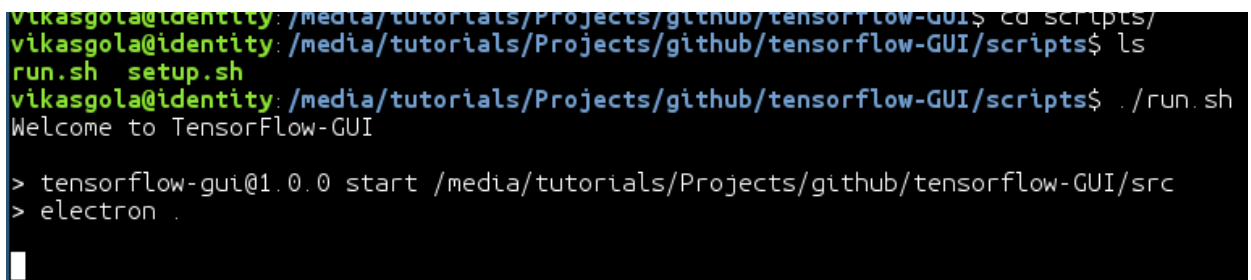
- Ubuntu 16.04 with [Anaconda](#) (Recommended)
- Ubuntu 16.04 without Anaconda (Experimental)
- Ubuntu 17.xx (Experimental)
- Ubuntu 18.xx (Experimental)
- Ubuntu 19.xx (Experimental)

C. Installation Procedure

- Clone the repo from GitHub
`'git clone https://github.com/sharmalab/tensorflow-gui'`
- Give permission to scripts to install the required libraries
`'cd tensorflow-gui/scripts/'`
`'chmod +x setup.sh run.sh'`
- Run the script to setup and install required libraries
`'./setup.sh'`

D. Development Procedure

1. Start the TensorFlow-GUI from *scripts* folder using command `./run.sh`

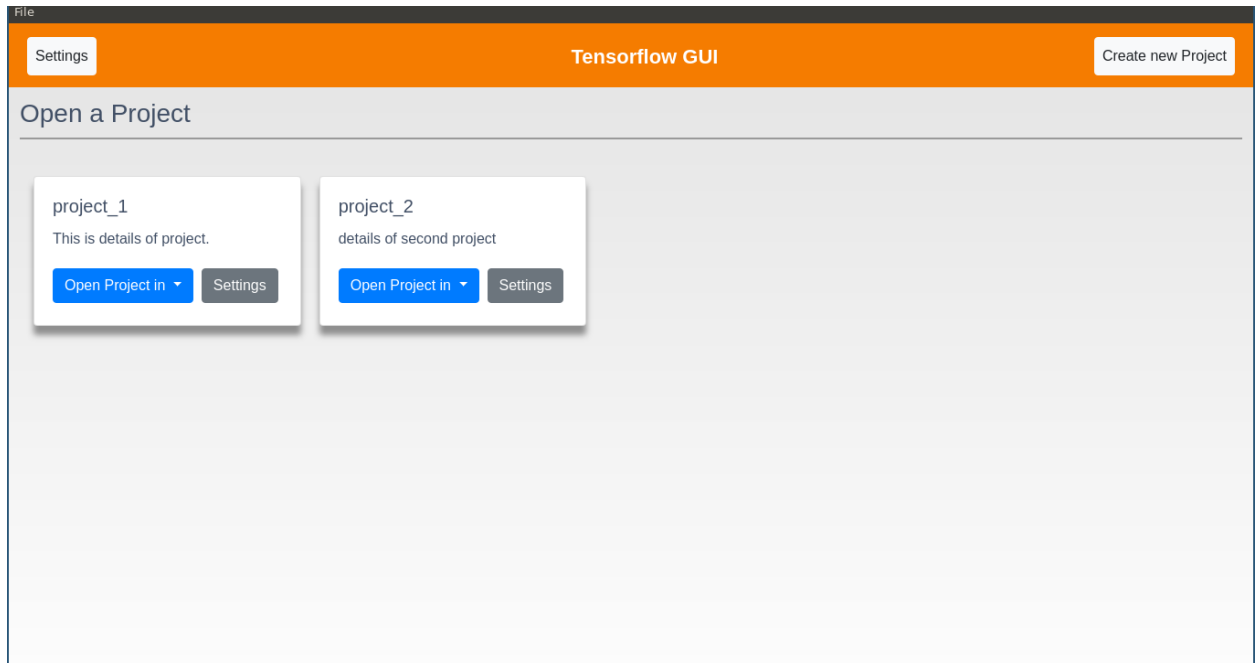


```
vikasgola@identity: /media/tutorials/Projects/github/tensorflow-GUI$ cd scripts/
vikasgola@identity: /media/tutorials/Projects/github/tensorflow-GUI/scripts$ ls
run.sh  setup.sh
vikasgola@identity: /media/tutorials/Projects/github/tensorflow-GUI/scripts$ ./run.sh
Welcome to TensorFlow-GUI

> tensorflow-gui@1.0.0 start /media/tutorials/Projects/github/tensorflow-GUI/src
> electron .
```

You would see page shown below in the screenshot. You will see all your created project on this page. You can open your projects by clicking on the button 'Open Project in' and you would get option to open it in code editor or node editor. Projects opened in node editor would lead to **loss** of all your custom code in code editor and new code would be generated after you generate your model.

If you find something wrong, then please report about it with screenshot or details on the GitHub page of TensorFlow-GUI.



To go to projects settings page, click on the Settings button in respective project card. New Project settings page will open where you could change details of project and also can delete the project.

To create a new project, click on the 'Create New Project' button located in the right side of the header. You would be asked for the name and details of the project. Complete the steps and your project would be created.

2. *Opening a project in code editor* would open page shown below. New created project would contain a default example code in code editor to get started easily. You can edit the most of the code in the code editor. However, don't change the code where it's mentioned as it can lead to project broken or errors. Save button in the header could be used to save the project.

```

vikasgola@identity: /media/tutorials/Projects/github/tensorflow-GUI/scripts
tensorflow gui

File Edit View Window Help

hello_world
description...

Tensorflow GUI

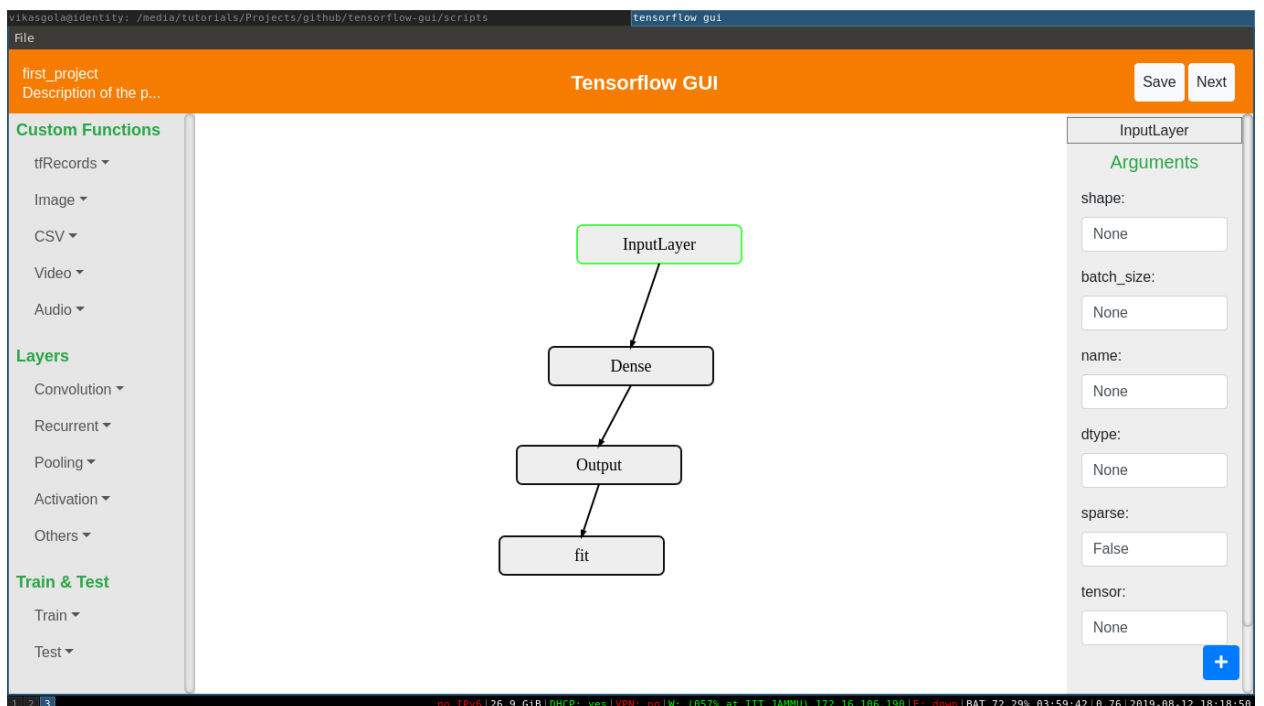
Save Train

129 Y = None
130 return X,Y
131
132
133 # Called Functions
134 X,Y, = LoadImageRecord(record_filepath = 'testing/record.tf',image_size = (28,28,3),batch_size = 128,shuffle = True,buffer_size = 1024,num_repeat = 100
135
136 tf.summary.image('input_data', X)
137 tf.summary.merge_all()
138
139 # Generated Model
140
141 def Network():
142     InputLayer_1 = Input(shape = None,batch_size = None,name = None,dtype = None,sparse = False,tensor = X,)
143     Conv2D_1 = Conv2D(filters = 32, kernel_size = (3, 3),strides = (1, 1),padding = 'valid',data_format = None,dilation_rate = (1, 1),activation = 'relu
144     Flatten_1 = Flatten()(Conv2D_1)
145     Dropout_1 = Dropout(rate = 0.25,noise_shape = None,seed = None,)(Flatten_1)
146     Dense_1 = Dense(units = 128,activation = 'relu',use_bias = True,kernel_initializer = 'glorot_uniform',bias_initializer = 'zeros',kernel_regularizer
147     Dense_2 = Dense(units = 10,activation = 'relu',use_bias = True,kernel_initializer = 'glorot_uniform',bias_initializer = 'zeros',kernel_regularizer
148     model = Model(inputs=InputLayer_1, outputs=Dense_2)
149     optimizer = tf.keras.optimizers.SGD(lr=0.001, momentum=0.0, decay=0.0,)
150     model = Model(inputs=InputLayer_1, outputs=Dense_2)
151     model.compile(metrics=['mae', 'accuracy', 'mse', 'mape', 'cosine', 'categorical_crossentropy'], optimizer=optimizer, loss = 'categorical_crossentropy
152     return model
153
154
155 # function for training model
156 def train():
157     model = Network()
158     x,y = getTrainingData()

```

OR

Opening a project in node editor would open a node editor page shown below.



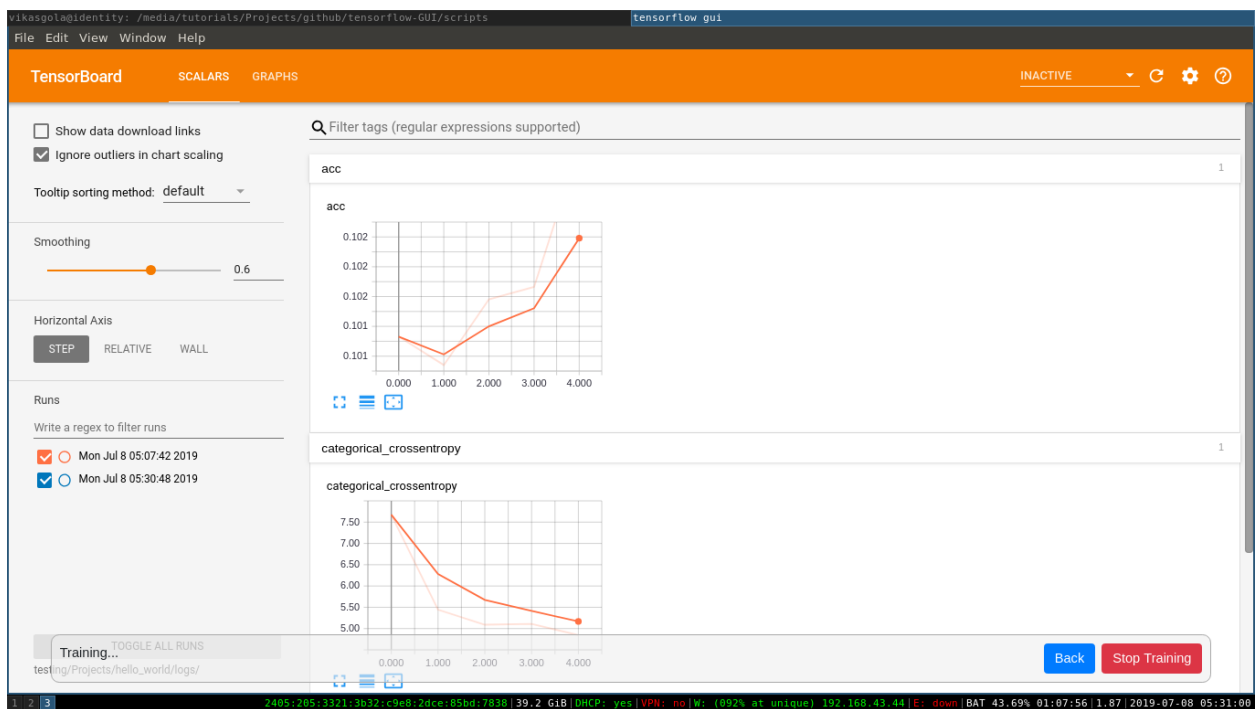
An example model is given in every new project to get started easily. You can delete any node of model on selecting it by left click and hit the delete button. Now, To

create new nodes you can go through left bar and select a layer and drag & drop it to the container. To join layers right click on a node (after this it would start showing an arrow from this node to your cursor) and then right click other node. Left click any node and it will open a right bar which have all the return variables and parameters of the node which could be edited. When you are ready, hit the save button and click 'Train' button to start training.

Note: For better understanding, go through the example videos. Links are given at the end of the document.

E. Training Procedure

To start the training click on the 'Train' button on the code editor page which would test if generated code is correct and then start the training of the mode. Training progress could be check live here on tensorboard.



To stop the training in the middle, click on 'Stop Training' button. Or you could wait for the training to complete. Status of the training can be seen at the left of the control bar. To

go back to code editor, 'Back' button in the control bar could be used. Note that you can only go back if training is finished or stopped.

F. Testing Procedure

Testing of the models could not be done in this version of product as it requires a totally different module to be built which would be developed and added in the upcoming versions. Till then you could use generated code from TensorFlow-GUI and edit it to test the trained model.

Methodology & How it works!

Tensorflow-GUI project has all source file of the software in the folder named *src*. The project is built using [electron](#) which is an open source framework and uses web technologies to build desktop applications.

- After starting the software, *src/index.js* script runs which creates a window and loads a default html page located at *src/modules/index.html*.

src/index.js

```
...  
// Creating main window of software  
win = new BrowserWindow({  
  width: 800,  
  height: 600,  
  nodeIntegration: true  
});  
// loading the root module or page  
win.loadFile('modules/index.html');  
...
```

- *src/modules/index.html* is kind of root of our project in which every required css is linked and contain main layout of the software. It contains a div element in which all modules html is loaded and replaced whenever required.

Every html page in module contains a specific css file and js file for it with the same name of html file. This root html also has css and js file for it at locations *src/modules/index.css* and *src/modules/index.js*.

src/modules/index.html

```
...
// load and show this loading gif whenever there is some process of
loading new module
<div id="loading" style="color:black">
  <p>Loading. Please Wait...</p>
  
</div>

// main div element where modules are loaded
<div id="main-content"></div>
...
```

src/modules/index.js

```
// function for loading html pages by path
function loadPage(page_path) {
  $("#main-content").html("");
  $("#main-content").load(page_path);
}

// loading user module
$(document).ready(() => {
  loadPage("user/user.html");
});
```

- The user module is located at *src/modules/user* which is default page or home page of our software. This module's html page located at *user.html* mainly contains list of all projects which is loaded using javascript in *div* element with id *user-projects-card-row*.
 - *loadProjects* function in *user.js* finds all the created projects and append as card in the page.
 - *openProject* function in *user.js* open a project in Node Editor or Code Editor whichever request has been made by user by loading the *src/modules/nodeeditor*

and *src/modules/codeeditor* modules respectively. It also starts a child process of tensorboard by first killing all already running tensorboard processes.

- Settings button with is `user-settings-button` load and opens the module *src/modules/settings*. Project Settings button load and opens the module *src/modules/project*.
- The Code Editor module is located at *src/modules/codeeditor* and loads the project code in code editor. It's *codeeditor.html* html page contains a div element which act as an editor.

[src/modules/codeeditor/codeeditor.html](#)

```
...
// div element in which code editor is loaded
<div class="draw-content">
  <div id="draw-main">
    <div id="code-editor"></div>
  </div>
</div>
...
```

[src/modules/codeeditor/codeeditor.js](#)

Below code creates text editor using [CodeMirror](#) library and loads the saved code in text editor.

```

...
// Creating code editor using CodeMirror library in div element with id
code-editor
var codemirror = CodeMirror(document.getElementById("code-editor"), {
  mode: {
    name: "python",
    version: 3,
    singleLineStringErrors: false
  },
  lineNumbers: true,
  indentUnit: 4,
  smartIndent: true,
  styleActiveLine: true,
  matchBrackets: true,
  value: global.editorText
});
...
...
// Loading saved code and load to text editor
global.editorText = fs.readFileSync(basepath + dir + "/editor.py", "utf8");
codemirror.setValue(global.editorText);
...

```

On clicking train button on code editor page, `testPython` function is called which try to compile the python code and reports the error and if there is no error it loads the *training* module.

- The Node editor module is the most important module which have the gui for creating model using nodes. It also depends on many js library files which are located at location *src/lib/*. This module is located at *src/modules/nodeeditor*.

```

function createLabel(x,y, text, layertoadd, id=null){
    ...
    //this function creates a node in the container and returns it
    ...
}

function addArrow(node1,node2, layertoadd){
    ...
    //this function creates a arrow between two nodes in the
    container and returns it
    ...
}

function saveProject(){
    ...
    // saves the project files in project folder whenever user
    click on save button or user goes to next section for developing the
    project
    ...
}

```

- The Training module is the module where deep learning models get trained and training status is updated. This module is located at *src/modules/training*.
<src/modules/training/training.html>

```

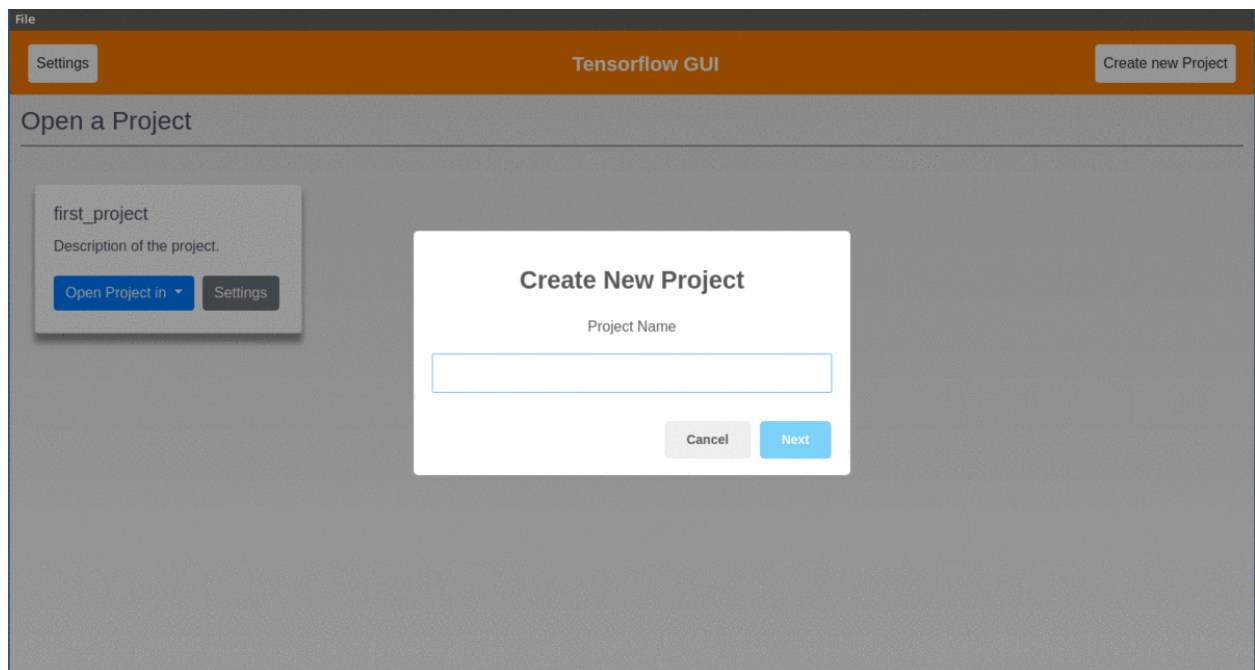
// iframe loads the tensorboard and div with id control-bar contains a
bar for some controls
<div id="training-main">
  <iframe id="tensorboard" src="http://127.0.0.1:6006"></iframe>
  <div id="control-bar" class="controls">
    <p id="training-status" class="d-inline">
      Starting Training...
    </p>
    <button id="stop-button" class="btn btn-danger ml-1">
      Stop Training
    </button>
    <button id="backPage" class="btn btn-primary mr-1">
      Back
    </button>
  </div>
</div>
....

```

[src/modules/training/training.js](#)

```
...  
...  
// runs python code which starts training the model  
// Refresh the tensorboard every 8 second till end of training by  
clicking the reload button  
$(document).ready(function () {  
    $("#training-status").text("Starting Training...");  
    runPython();  
    intervalid = setInterval(() => {  
        $('#tensorboard').contents().find('#reload-button').trigger("click");  
    }, 8000);  
});
```

Other Examples and Snapshots



Other example videos can be found [here](#):