Karthik Ramesh Iyer
Github: KarthikRIyer
Email: kiyer@ch.iitr.ac.in
Skype: live:karthik.iyer2_1
Gitter nickname: KarthikRIyer
Phone Number: +91 84606 69619
Time zone: UTC +05:30
Town, country: Roorkee, India

TensorFlow GSoC 2019 Proposal

A library for Data Visualization in Swift

Meeting with mentors

- Before 10 May : Reachable anytime between 3:30 am to 7:30 pm (UTC) [9:00 am to 1:00 am IST] through Email/ Gitter.
- After 10 May : Reachable anytime between 4:30 am to 8:30 pm (UTC) [10:00 am to 2:00 am IST] through Email/ Gitter.

  Can join a planned video session if required.

Abstract

Swift for TensorFlow is a Swift library that helps develop and train ML models. Data Visualization can be helpful when exploring a dataset. It helps in identifying patterns, corrupt data, outliers, etc. It helps in the qualitative understanding of data. But as of now, there is no way one can plot graphs natively using Swift that works cross-platform. We do have CorePlot, but it works only on macOS and iOS. This project aims to make a Data Visualization library (similar to matplotlib) for Swift that works on Linux, macOS and Windows as well.

Project Ideas

The key plots necessary for basic Data Visualization are:

- Line Chart
- Bar Chart (vertical and horizontal)
- Histogram
- Scatter Plot

Other features planned for this project are:

- Contours
- Images
- Fields
- Subplots

Implementation details

The implementation will include three main layers :

- **Developer facing external API** : Using this layer the developer will provide data to the plot. This layer will be written in Swift.
- **Plot generating layer** : This layer will get the data from the top layer and will generate the actual plot in terms of vector elements. All the logic to generate plots will be contained here. In the end it will pass only the points required to generate the plot to the third layer. This layer will be written in Swift too.
- **Rendering layer** : This layer will have minimal to no plot generation logic. It will have the functions to draw primitives such as lines, rectangles, text, etc. I aim to implement two rendering backends for this project. One will be the Anti-Grain Geometry(AGG) rendering library. In case I am able to get access to a Mac I'll implement CoreGraphics. Otherwise I'll implement a **simple native-Swift SVG renderer**. Here are the SVG specifications for reference: SVG

There will also be a bridging layer between the C++ and Swift Code that will allow both to interact.

The above design will provide us with the flexibility to implement multiple rendering backends that can take advantage of the platform they're running on. For example, if we would like the plot

generation to be hardware accelerated we can use OpenGL, or if the library is being used on macOS, we can have CoreGraphics or Metal as the backend.

Other dependencies :

- lodepng : LodePNG is a PNG image decoder and encoder. This library will be used to read images, and save plots in the PNG format when required. When most of the important features are implemented in the library, we can also remove this dependency and write the code to save the images.

All the above dependencies will be wrapped into packages using the Swift Package Manager.

The primary goals for this project are:

Line Chart

The desired features for this plot are :

- Plot **y** vs **x** when both are given as arrays
- Plot **y** vs array index when only **y** is given
- Allow plotting multiple sets of data and show legends
- Allow setting colors of each plot
- Multiple independent **y** axes
- Allow plotting functions like sin(x) , cos(x), etc

Bar Chart

The desired features for this plot are :

- Plot bars using **y** and **x** values given
- The given **x** might be numeric or non numeric
- Allow setting colors of each bar
- Stacked Bar Chart
- Allow hatching of bars

Histogram

The desired features for this plot are :

- Plot histogram from **x** and **y** arrays
- Plot histogram between any two **x** values with **y** as a function of x
- Allow setting alignment of bars with respect to bins : left, mid (default), right
- Allow setting histogram color
- Allow plotting of different types of histograms:
    - bar (default)
    - stacked histogram
    - step : generate unfilled line plot
    - step filled : generate filled line plot

Scatter Plot

The desired features for this plot are :

- Create a scatter plot from **x** and **y** arrays
- Allow setting either single color or an array of colors for the plot
- Allow creating multiple plots with legends
- Allow setting marker shapes

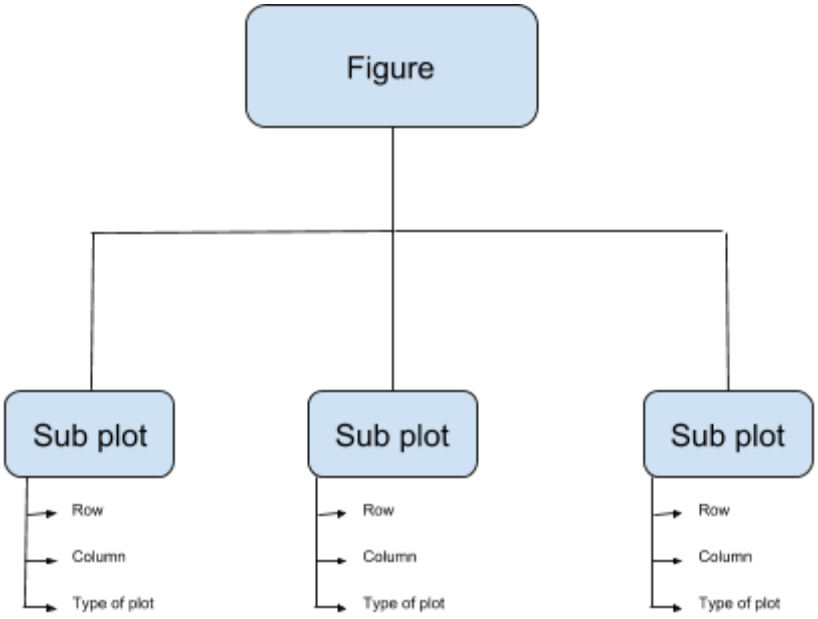
Images

The desired features are :

- Read and display images, with or without axes.
- Allow writing a rasterized bitmap.

Sub Plots

This will allow the developer to create multiple plots in one figure. Each plot may be of a different type. The structure can be as follows:



Each figure will have an **array of Subplots**. Each Subplot wil have a row and column property, and will contain a plot object. All the above described plots will inherit from a plot class containing common properties.

➔ I also plan to allow writing rasterized bitmaps so that the bitmap may be displayed in Jupyter/Colab notebooks using Python code. Reference to Bitmap file specification: BMP

The secondary goals for this project are:

Contours

The desired features for this plot are :

- Allow the user to create **x** and **y** arrays using a function similar to meshgrid in matplotlib, then define a `**z**` variable depending on **x** and **y.**
- Plot contours with the above **x, y,** and **z** values.
- Allow setting the two extreme colors for the contour.
- Allow adding lines and labels to the contour.
- Allow changing the styles of the lines, i.e. solid, dashed, dotted, dash-dot.

Fields

The desired features for this plot are :

- Allow plotting Vector fields from the following parameters :
  - **x** array representing x-coord of arrows
  - **y** array representing y-coord of arrows
  - **u** array representing x-component of arrow vectors
  - **v** array representing y-component of arrow vectors
- Allow setting colors of the arrows.
- Allow changing the properties of arrow like their, thickness, length of head, etc.

➔ Another great addition would be to integrate directly with IPython kernel's display API for inline rendering in Jupyter/Colab notebooks.
➔ Implementing real time rendering using OpenGL/Metal would be a great addition, something that CorePlot lacks at the moment.
➔ Explore and implement methods to update plots as data is added.

Milestones

Milestone 1 *(Deliverable before Phase 1 Evaluation)*

I'll start by writing the basic functions required in the renderer to draw primitives. I have already written roughly the functions for basic primitives like lines, text and rectangles for AGG. This will need to be done for SVG/CoreGraphics as well. Other primitives like arrows, and shapes for markers will be implemented later on. Then I'll move on to implementing the basic plots. The plots I'll cover in this period are Line Chart and Bar Chart. Implementing Line Chart and Bar Chart shouldn't take much time because the logic for these plots can be lifted from the library I've developed for Android, Graph-Kit.

Milestone 2 *(Deliverable before Phase 2 Evaluation)*

I'll implement Scatter Plot and Histogram during this period.

Milestone 3 *(Deliverable before Final Evaluation)*

During this period I'll implement Image Support. I'm keeping less workload for this period, so that it compensates for any unexpected delays I might encounter.

Milestone 4 *(Wishlist, If time permits)*

In case I'm able to complete the first three milestones within time I'll move on to implementing the secondary goals, stated in the implementation details.

Here is a more detailed timeline :

Present - April 9 (Homework period)	Get more familiar with data visualization, explore ways to create different plots.
April 10 - 5 May (Hiatus)	End semester examinations will be near so I'll be inactive, but will be available for communication through Email/Gitter
May 6 - May 12	Community bonding period: Discuss with the mentors any important points missed and plan the work ahead
May 13 - May 26	Write renderer backends for plotting Line Chart
May 27 - June 2	Write developer side API for Line Chart, link all layers and document the above part
June 3 - June 12	Write renderer backends for plotting Bar Chart
June 13 - June 17	Write developer side API for Bar Chart, link all layers and document the above part
June 18 - June 24	Write renderer backends for plotting Histogram
Phase I Evaluations	**Milestone 1 reached**
June 25 - July 4	Write the developer side API for Histogram, link all layers and document the above part
July 5 - July 15	Write the renderer backend functions for Scatter Plot.
July 15 - July 22	Make the developer side API for Scatter Plot, link all the layers and document it

Phase II Evaluations	**Milestone 2 reached**
July 23 - July 26	Write the renderer backend functions for image support
July 27 - August 4	Make the developer side API for image, link all the layers and document it
	Milestone 3 reached
August 5 - August 12	Work on integrating IPython kernel's display API
August 13 - September 2	Work on supporting OpenGL as a rendering backend
August 19 - September 2	Do a general code cleanup. Make sure there is nothing left undone and everything is tidy. Prepare for **Final Evaluation**
After September 2	Keep contributing by completing leftover secondary goals, and extending multiple rendering backends as necessary.

About Me

I am a Sophomore at **Indian Institute of Technology Roorkee**. I am majoring in Chemical Engineering. My areas of interest include **Computer Graphics, Creative Programming, Android Development,** and **Video Editing & VFX**. I developed a passion for programming in Grade 8, when I enrolled for course "Programming in Core Java". Later on in Grade 10, I enrolled for a course in Advanced Java. I've been programming ever since, but became aware of open-source in my freshman year at college. I've contributed to appleseedhq and OpenFoodFacts in the past. Currently I am a member of Mobile Development Group IIT Roorkee, a bunch of passionate enthusiasts trying to foster software development culture in the campus.

I've found Computer Graphics to be my passion, and have also observed that machine learning has become important in many fields. I've been meaning to explore this field for a while and this project seems to be a good entry point for me to learn how data visualization is done and is used in machine learning. I am really excited to work with the Swift for TensorFlow team and hope that this summer proves to be a great learning experience for me.

Coding Skills

- Programing Languages
    - Fluent in Java with a sound knowledge of OOP
    - C++
    - Swift (Familiar with the basics, will make myself proficient enough before the coding period)
- Development Environment
    - Atom text editor
    - CLion
    - OS: Ubuntu 18.04 LTS, Windows 10
- Version Control
    - Git

Why me for the project?

I've been studying Computer Graphics for a few months now. I haven't looked into 2D rendering yet. This project seems to have all the things I desire to explore. I'd come to know a bit about the algorithms used in machine learning and how their data needs to be visualized and will simultaneously learn about 2D rendering. I've already learnt a bit, while exploring the requirements for this project. Besides this, I've been developing for Android for almost two years now. As a result, I am fairly acquainted with the specifics of Java, OOP and Software Development Techniques. I am also familiar with C++, and have covered the basics of Swift. I am confident that I'll be able to polish my Swift skills before the GSoC coding period. I make atomic commits with clean commit messages and well structured PRs.

I can easily devote around 50 hours per week during my timeline. I believe that the allotted work per week is completely doable for me and is neither overloaded nor slacked. This makes me eligible to apply for the Data Visualisation project.

Other Commitments

I have my end term examinations from 24 April to 4 May. So I'll be inactive between 10 April to 4 May which is way before the official coding period. My vacations start on 10 May and end on 11 July, and the official GSoC period is from 6 May to 26 August. I can easily devote 45-60 hours a week until my college reopens and 30-40 hours per week after that. I plan to complete most of the work before my college reopens. During the vacations I have no other commitments and shall keep my status posted to all the community members.