Web Programming Final Report

Numpy – Visualization

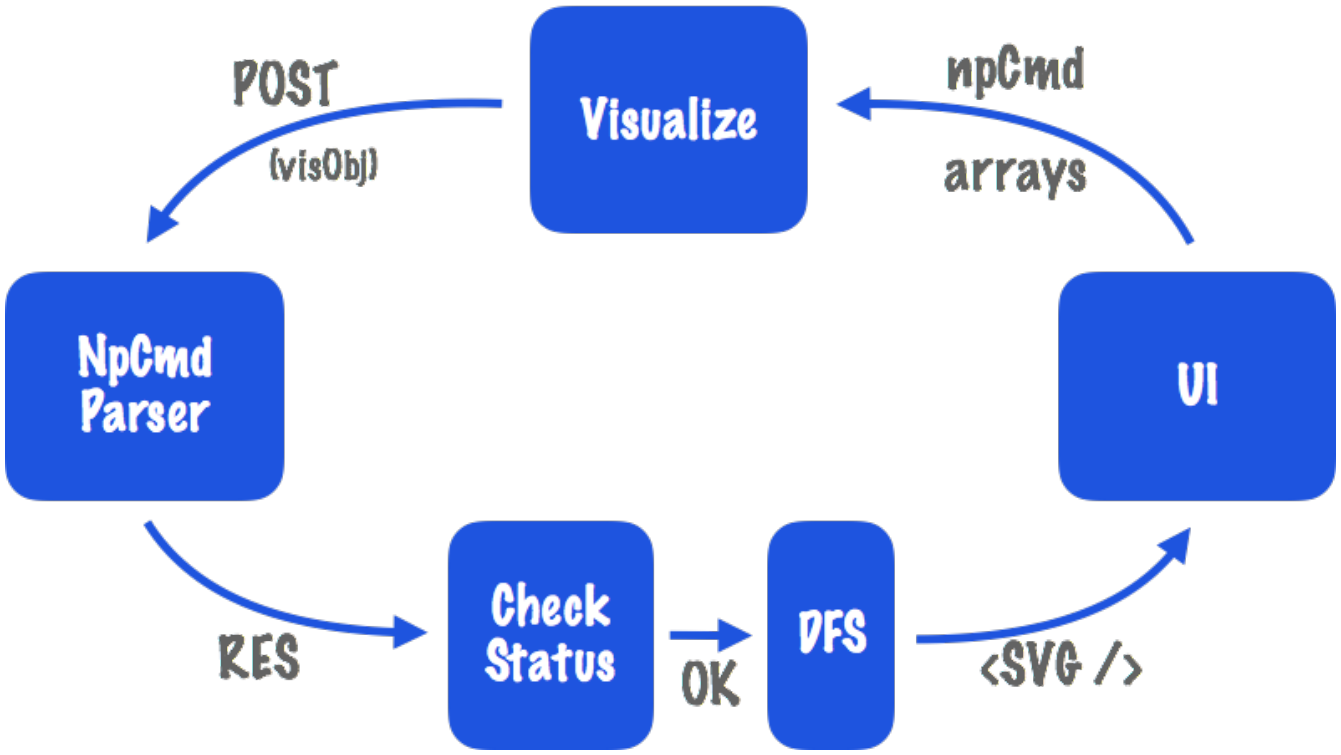
Group Number: 6
Group Members: r05943094 樊恩宇, r05921035 陳奕安
Contact: rock15688@gmail.com, r05921035@ntu.edu.tw

A. Introduction

For deep learning has gained so much attention in recent years, the Python package 'numpy' is widely used to cope with complex computation on multidimensional arrays, which is a significant part of deep learning. On grounds of that, we made a website that can visualize frequently used numpy commands. This website can help beginners understand how numpy handles and transforms multidimensional arrays.

We currently support these numpy commands: identity, ones, zeros, ones_like, zeros_like, reshape, transpose, swapaxes, tril, triu, **+, -, x,** /. For detailed description of these commands, please refer to the official documentation of numpy.

B. System Structure



After the user specifies the target numpy command "npCmd" (which may combine many numpy function calls) and predefined variables (name and shape), the information will be converted to a "visObj" and sent to "NpCmd Parser" on the server. If the parser successfully convert the visObj to abstract syntax tree (AST) and annotate each tree node with output shapes, it will responds with a JSON object with "result" being the annotated AST and status "OK". Finally, we traverse on the AST using DFS and visualize every visited numpy operation.

C. Technique

(1) Front-end - UI:

We use "Semantic-UI-React" to build up our pages. "Semantic-UI-react" is the official React integration for "Semantic UI". It is very flexible and contains many useful features,

making designing beautiful websites quickly. For more information, please refer to Semantic-UI.

(2) Front-end - Visualization:

"GSAP" is a general purpose animation library designed for front-end developer. Four core objects, "TweenLite", "TweenMax", "TimelineLite", and "TimelineMax", and their combinations can create stunning animations on HTML tags as well as SVG tags in just a few lines of code. Because it's SVG component that we are dealing with, we hardcorely calculate transformations parameters for the animations (e.g. x-offset) ,and create a TweenMax object for each supported numpy operation in the requested command.

The responded object from the server is a tree structure with information about the command to be visualized. Each "node" inside the object has a property called "children", which gives us the information to traverse through the object by DFS. Then, according to the order of the DFS traversal plus the "timeStep" information we calculated, we update component state every 5 seconds on visualization start and render each node using TweenMax object mentioned above.

Here we have to admit that many trade-offs are made due to a tight development schedule. SVG is not a good option when responsive web design (RWD) feature are taken into account. In that case, canvas may be a better technology, and a good canvas animation library would be used to keep animations in sync with React lifecycle. However, to the best of our knowledge, no open-sourced library exists to make the implementation of our animation component extensible to match a variety of numpy operations, and therefore "GSAP" is our best hope before deadline.

(3) Back-end - Server & Parser:

The action of parsing a numpy command and returning a desired tree structure can be described as a two-staged operation. Firstly, a numpy command is parsed into python AST. In this stage, we only get general syntax information and no domain-specific features are generated. "ast" and "asttokens" are utilized to parse numpy commands. Secondly, multiple operations occur as we traverse AST by DFS. Plain numbers and tuples will be used to calculate output shape (just like compilers do constant folding). Each node will have its type and output shape annotated. Arguments number and type will be checked to make sure it is in compliant with the usage of numpy functions. After these operations are done, a JSON object is generated to be sent to client-side.

The implementation of the server is very straightforward. We use "flask" to construct (the only one) RESTful entry point. NodeJS is not suitable in this case because we don't want to maintain the connection between our server and parser. Our front-end application is served as static files, while the entry point receive "POST" action with numpy command and respond with the generated JSON object. The implementation of the server only consists of one file with ~150 lines.

D. Usage

The usage of our website is pretty easy. At the “Visualization” page, Users can predefine arrays with “name” and “shape”. Then, they can specify the command to be visualized using the defined array. To be noticed, we restrain array and command declaration to be exactly the same as using numpy in Python, because we want to make sure the users do know that they are visualizing numpy not other package. Finally, click the button “visualize”, and the animation will begin. For more detailed usage, please refer to the figures at the last section of the report or the home page of our website.

E. Future Work

(1) Add More Commands:

Add more frequently used numpy commands in deep learning application to be visualized or some more complicated command can be added in.

(2) Extend Visualization:

Extend visualization to allow multiple commands on one request. Users can write commands in in-browser text editor and each line will be visualized in order.

(3) Extend to Other Python Libraries:

Extend this visualization framework to other popular libraries like Keras and Pandas. We should create a rewritten, standalone parsing library to prevent duplicate code.

(4) Linked with Database:

Maybe set up a login system for different users to organize their preferences on visualization.

F. Collaboration

樊恩宇: Front-end, Visualization(DFS), Report.

陳奕安: Back-end, Visualization(GSAP), Report.

G. Review

(1) 樊恩宇 :

For me, this is the first time I expose myself to Web related course. Before this semester, I know nothing about the world of Web. Hence, this final is a great challenge for me, and I am very lucky to have such a powerful partner to collaborate with and learn from.

I learned a lot from this project, like how to use “Semantic-UI-React”, more clear to relation between state and rendering…etc. Although these may sound like extremely

fundamental stuff of web programming, this is my first "bigger" website, at least bigger than homework, so I gain many from it. Furthermore, during the final, what I consider to be one of the most important thing is the "unit testing", since you never know what the user will do to your website whether or not you've specified all the rules. Especially when writing constraint to the user inputs.

Last but not the least, I want to appreciate all the instructors of this course for giving such a substantial and fruitful class this semester. I know I still know little about web programming, and as mentioned in the courses, this field is alternating tremendously fast, so, one should always enrich themselves constantly and should always be prepared for the incoming challenges.

(2) 陳奕安:

The idea of calculating output dimensions for NumPy operations bring to mind when I took MLDS course last year as a beginner in both Python and NumPy. But it's not until the website "Promisees" was mentioned as we learned Promise in the class did I realize that visualization can be friendly to beginners. I hope this project can really help others in some way.

All of the techniques except "Python AST" have been used on other projects of mine: I learned how to use "GSAP" when creating a website for EE Summer Camp, started using unit testing framework in this class (really appreciate that!!), and became acquainted with Python since it has accompanied me for countless, painful "buggy" nights. Many people might think that there are so many technologies to learn while much more are spawning on the way. Rather, we should see every project as an opportunity to learn new technologies. The efforts of reading docs of libraries and the experience of debugging programs on tricky cases will prepare yourself for the next bigger challenge. I hope I can learn how to cooperate with others in a more "open sourced" way when revising the code in the future.

Last, I want to thanks all the instructors in this course. The tools mentioned, the libraries used, and the principles followed are just options suitable for our current need, and these options may change. What won't be changed are the taste of good project structures, the good habits of writing code, and the importance of testing, etc. These are conveyed implicitly or explicitly during the class and help me a lot. Thank you all for the unchangeables.

H. Github URL

Web Programming Final Repo

I. Website's Figure

Figure 1:

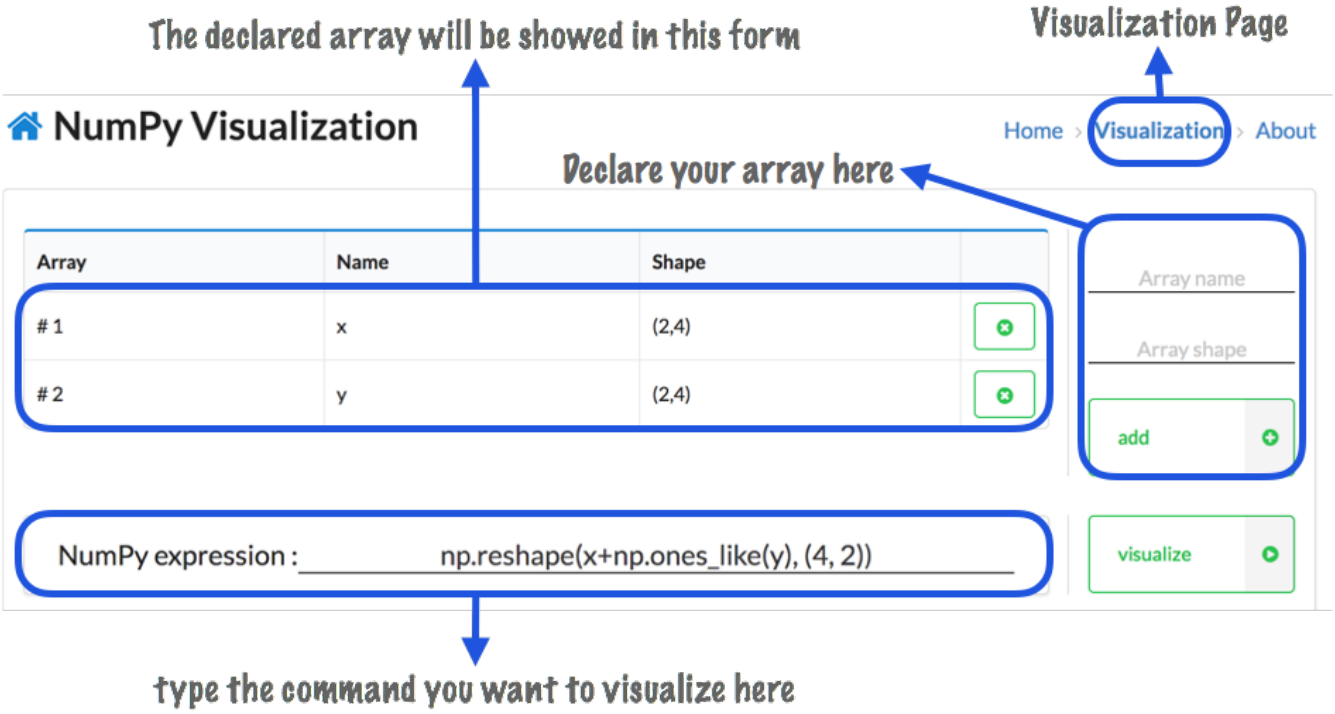


Figure 2: