COMS W4156: Preliminary Project Proposal

Team: Wall-E
Members: Yao Fu (yf2470), Wufan Shangguan (ws2541), Qing Teng (qt2126), Wenqi Wang (ww2505)

Project Name
LrnDeep

0.1. Language and Platforms
Language: Python 3.6, JavaScript, HTML
// Operating system: macOS Mojave & High Sierra
Platform: Web
Framework: Django 2.1.2
Database: SQLite 3.24.0

0.2. Technologies
Git 2.14.2: version control
Django 2.1.2: python web framework
Unitest: python unit testing framework
Keras 2.2.4: python deep learning library
Pip 18.0: python package manager
Pickle: python object serialization
SQLite 3.24.0: database program
GoJS: JavaScript library for building interactive diagrams

1. Synopsis
We aim to develop an interactive and straightforward method of building neural networks. Using our platform, generating functional deep-learning code is as easy as playing with Lego. Students and educators can utilize this platform to learn or teach deep learning in a simple, intuitive way. On the other hand, professionals can make use of this platform to visualize the structure of their neural nets, making it convenient to present their work to clients. Ultimately, we hope to avoid steep learning curves and make deep learning feasible for everyone.

In this platform, different layers of a neural network are represented visually by simple geometric shapes. Users can design their own neural networks by simply clicking and dragging, without having to deal with overly technical details. At the same time, the platform will generate code in real time that corresponds to the desired neural network. Users can easily update or modify a network in progress, and modifications will be visible both visually and also in the code generated. Furthermore, they can save their neural networks and restore their progress the next time they use our platform.

2. User Stories

1. As a student starting to learn deep learning, I want to familiarize myself with the process of building a deep neural network so that I can gain more understanding of the subject. My conditions of satisfaction are: 1) components appear on the canvas when I drag them from the panel; 2) components are removed from the canvas when I delete them; 3) hints appear when hovering on components or the canvas; 4) warnings are given when potential errors or points of confusion arise.

2. As an instructor, I want to generate deep learning code in real time so that I can explain the code to my students. My conditions of satisfaction are: 1) code appears in real time when I drag a component onto the canvas; 2) related code is removed when I delete a component; 3) graph and code can be easily saved and restored in order to continue from a previous class.

3. As a deep learning engineer, I am working in a team of a variety of backgrounds. I want to explain to my colleagues about the mechanism of neural networks so that they can quickly gain a direct impression of the tech parts. My conditions of satisfaction are: 1) different components of the neural network are represented in a straightforward way through different shapes and/or color; 2) the neural network graph can be exported as an image file for presenting to an audience.

3. Testing

Input	**Expected Results**
The user drags a geometric shape (representing a fully-connected layer) onto the canvas.	A geometric shape appears and remains on the canvas. The default properties of this layer (e.g. default input size) are listed by the geometric shape.
The user removes an existing geometric shape from the canvas.	The geometric shape disappears! :-)
The user hovers their mouse over a geometric shape.	A hint appears by the geometric shape (e.g. “This represents a sigmoid layer. It

	performs the sigmoid operation on its input.”).
The user drags a fully-connected layer with input size 64 and connects it with an input layer with size 128.	An warning message appears by the fully-connected layer to warn the user of a potential error.

2.

Input	**Expected Results**
The user finishes dragging a component onto the canvas.	The corresponding code subsequently appears in the code viewing panel.
The user deletes a component from the canvas.	The corresponding code subsequently disappears in the code viewing panel.
The user saves a working session.	The current canvas gets stored in the database.
The user loads a previously saved session.	The corresponding canvas and code are generated.

3.

Input	**Expected Results**
The user uses different shapes on the canvas.	The shapes appear in distinct colors on the canvas.
The user exports a canvas.	A corresponding image file is generated and downloaded onto the user’s device.