

# CSE 564 Visualization Final Project Preliminary Report: Visualizing Explainable Deep Convolutional Neural Networks

Bo Cao

SBU ID: 112130213

[bo.cao.1@stonybrook.edu](mailto:bo.cao.1@stonybrook.edu)

## Introduction

Nowadays, convolutional deep neural networks achieve state-of-the-art performance on image classification tasks. However, it is often treated as a black box that does well on the classification tasks, but lacks the interpretability for people to understand how it makes decisions. In this project, firstly I review the recent approaches that researchers and developers use to visualize deep learning models, then I point out the problems behind these methods to visualize the models and propose my method to visualize the deep learning model ResNet50 on MNIST dataset. The only task for the deep learning model here is image classification.

This project will be beneficial to many users, including developers, educators, domain experts and so on. For developers, it can help understand the model and thus know the hints to fasten the improvement of the model. For educators, it helps to teach students how the deep learning model works. The visualization of the explainable model helps build the trust between experts and AI system.

## Background

Recently, there has been research on interpretability of deep learning models. To address the question of why we need this kind of interpretability, Mueller et al. [1] summarize the important points of it: 1) *Verify that the classifier works as expected*; 2) *Improve classifier*; 3) *Learn from the learning machine*; 4) *Interpretability in the sciences*; 5) *Compliance to legislation*. In another talk [2], he also points out the fact that models that have similar performance could have different ways to make the decisions for classification - one is looking at the object in the image while the other is looking at the artifact. This leads to the question that which model should we trust. When it comes to autonomous car system and cancer diagnosis on medical images, experts often need to understand the rationale behind these deep models.

In terms of the techniques for visualizing deep models, recently there have been many implementations. One of the most popular visualization tool is TensorBoard [3], which can visualize a deep learning model simply by log file. Playground [4] is intuitive to visualize the data flow in the model. TensorSpace.js [5], the work from Harley et al [6] [7] and [8] are powerful tools to visualize the deep learning model in 3D. TensorFlon.js [9] is a convenient tool to deploy ML application on a browser. PyTorchViz [10] can visualize the executed paths in PyTorch. VisualDL [11] can visualize deep learning process and result. Keras provides its visualization tool [12] [13]. Other visualization tool includes netron [14].

One github repository iNNvestigate neural networks [15] uses different kinds of methods to visualize the relevance the network when deciding a class. One of the useful algorithm to decompose deep learning model is layer-wise relevance propagation (LRP) [16], which will be implemented and visualized in this project.

## **Problem**

Most of the recent methods for visualizing deep models mainly focus on two aspects: 1) showing the graph of the model and 2) displaying the features maps in the intermediate layers. This is helpful for us to understand the model, but it can only give us a general sense of the how the intermediate layers convolve with the corresponding inputs. To better understand the deep learning model, we need to understand how network makes the classification decision given an image. A key point to address this problem is to show clear evidence in the network that makes the decision.

Lots of efforts to visualize and understand the network have been put on displaying the feature maps in the intermediate layers [17] [18] [19] [20]. However, very often these visualization can only explain a lower level features from the model such as what kinds of edges some filters are looking at, which is not sufficient to interpret how the model decide which class.

Therefore, in the next section I will talk about the approach to address this visualization problem.

## **Approach**

In this project, I will implement the visualization from two aspects: 1) coding and 2) algorithms.

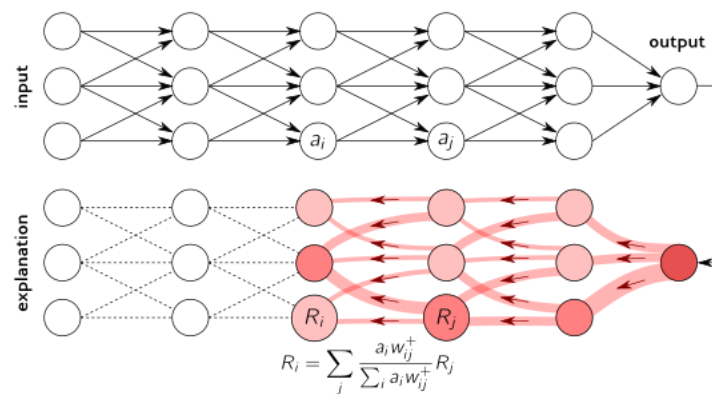
For coding, FLASK is used as the framework of the project. Python will be used as backend, D3 is used as frontend for visualization. Bootstrap is used simply as a template for the design theme. Regarding the framework for deep learning, Keras or PyTorch will be used depending on the progress of development.

The MNIST [21] will be used as the dataset in this project and the pretrained model of ResNet50 will be used as the backbone.

In terms of algorithms, first I will do t-SNE on the MNIST dataset and visualize it on the website, which maps all the images from a high dimension to a 2D dimension space. This will give user's a overall sense of the whole dataset.

The second part of visualizing algorithm is to visualize the deep learning model ResNet50, which is displayed as a graph that users can see each layer clearly.

Thirdly, I will focus on implementing the visualization of the Layer-wise Relevance Propagation (LRP) [22] [23], which is shown in Picture 1 [16] [24] [25]:

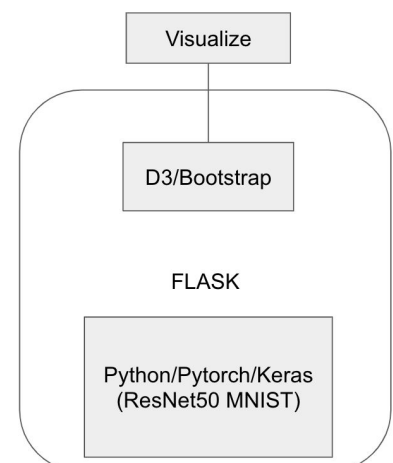


Picture 1

Note that the visualization rules such as choosing color space, deciding width length will be applied, and the tool stack may be changed during implementation.

Then, the whole architecture is shown in picture 2:

To my best knowledge, this is the first time to integrate and visualize all these algorithms in one implementation, which is the main contribution in this project.



Picture 2

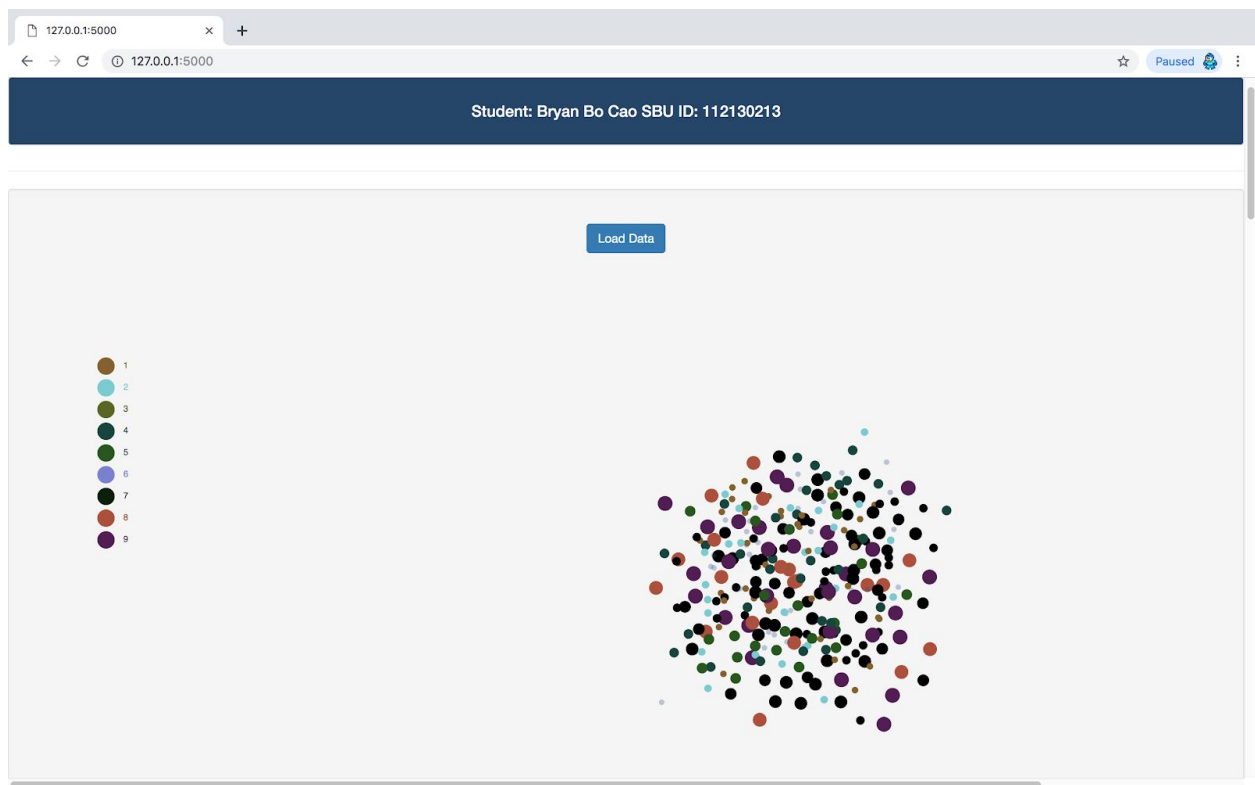
## Implementation

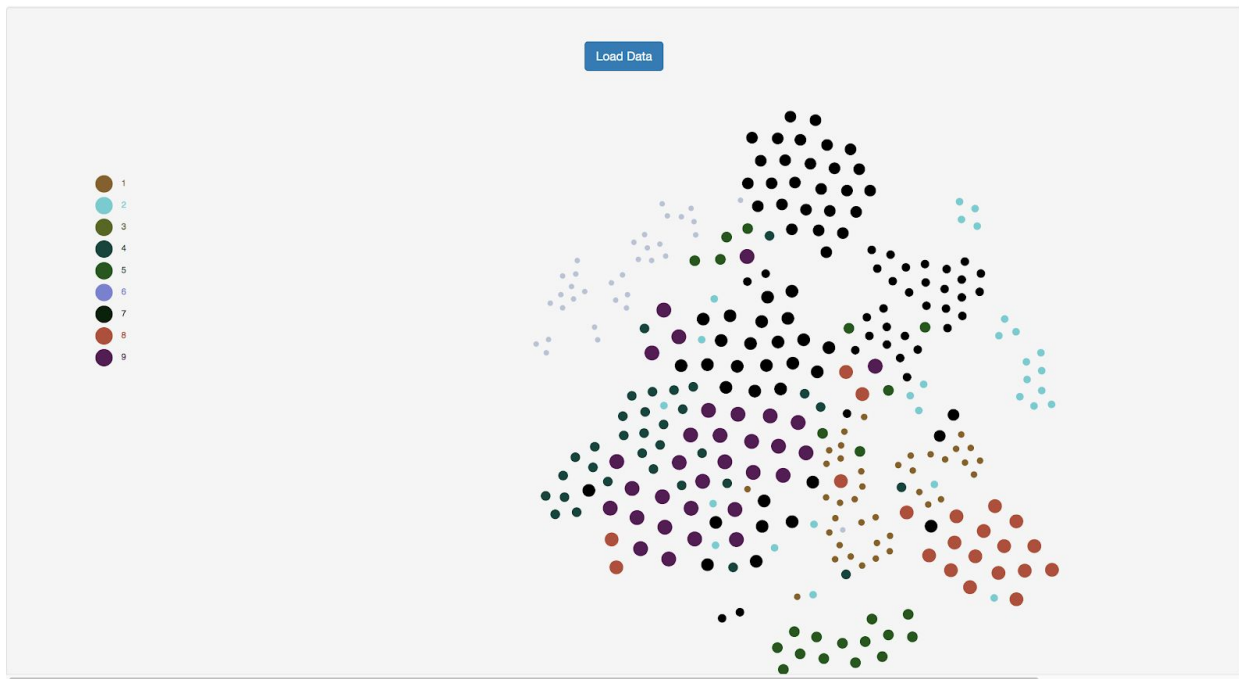
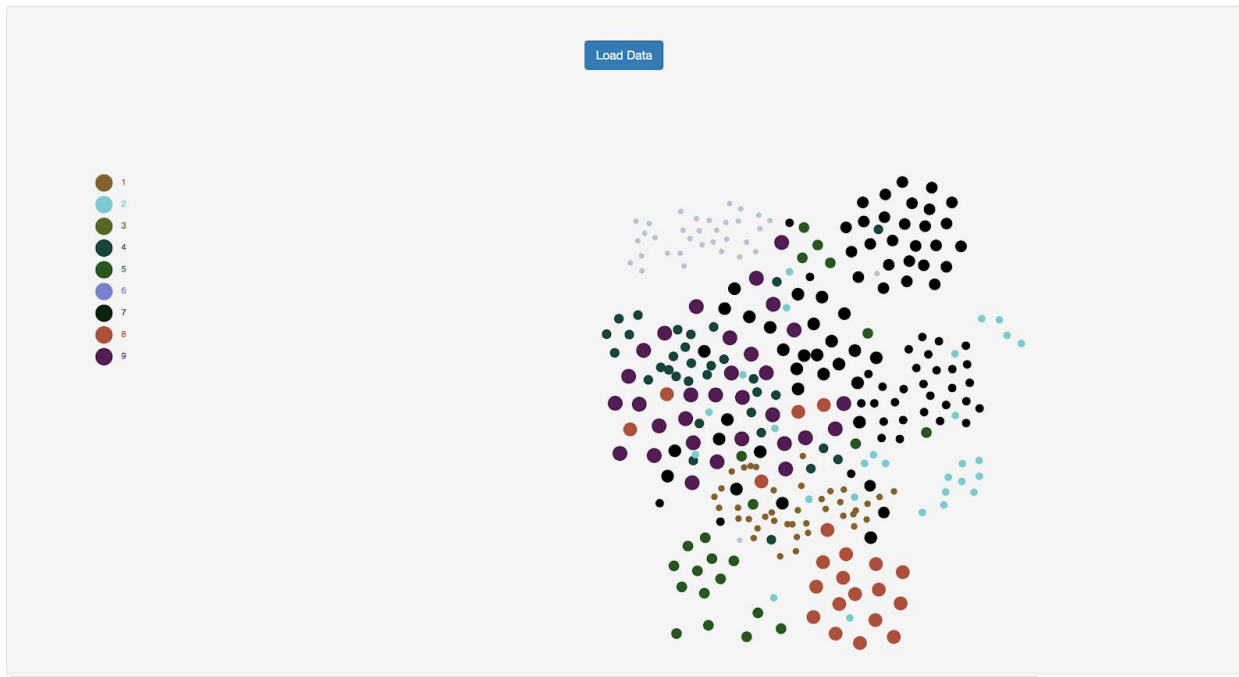
Below are the screenshots of the visualization process, including three main parts: 1) T-SNE on MNIST, 2) VGG-16 Architecture and 3) Activations.

### T-SNE on MNIST

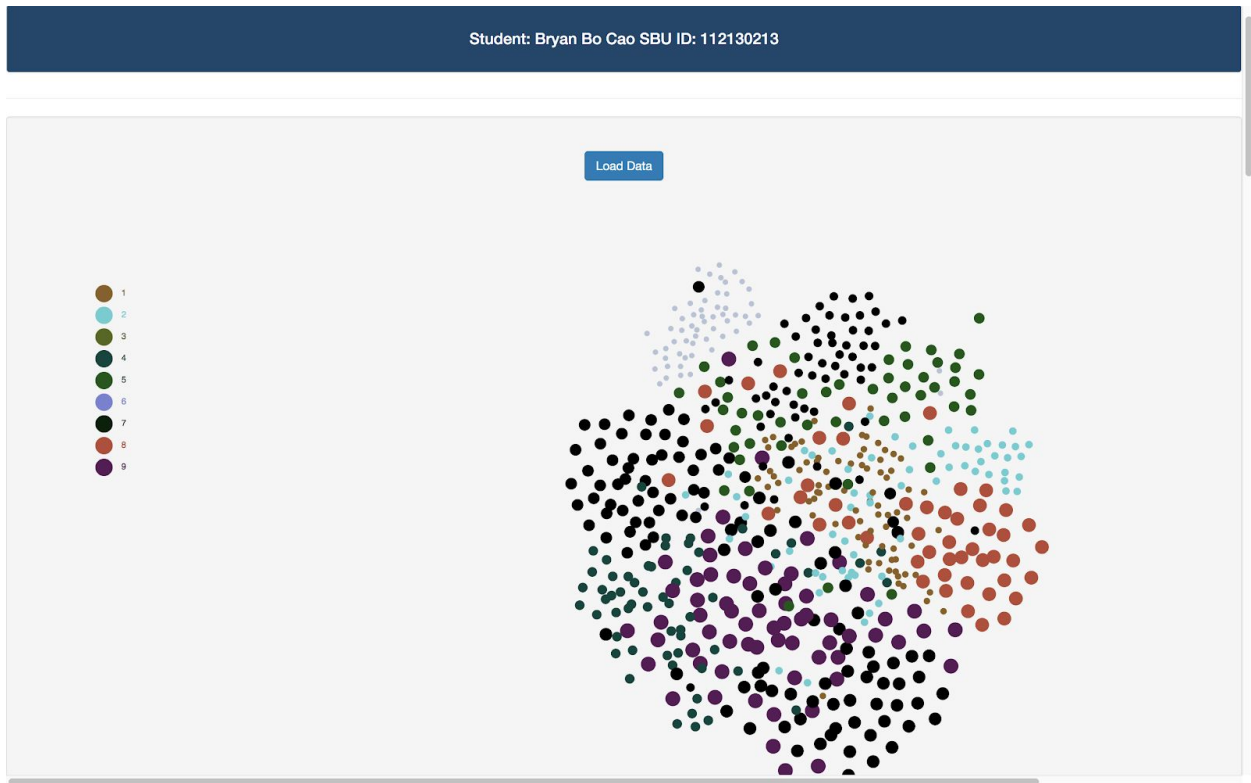
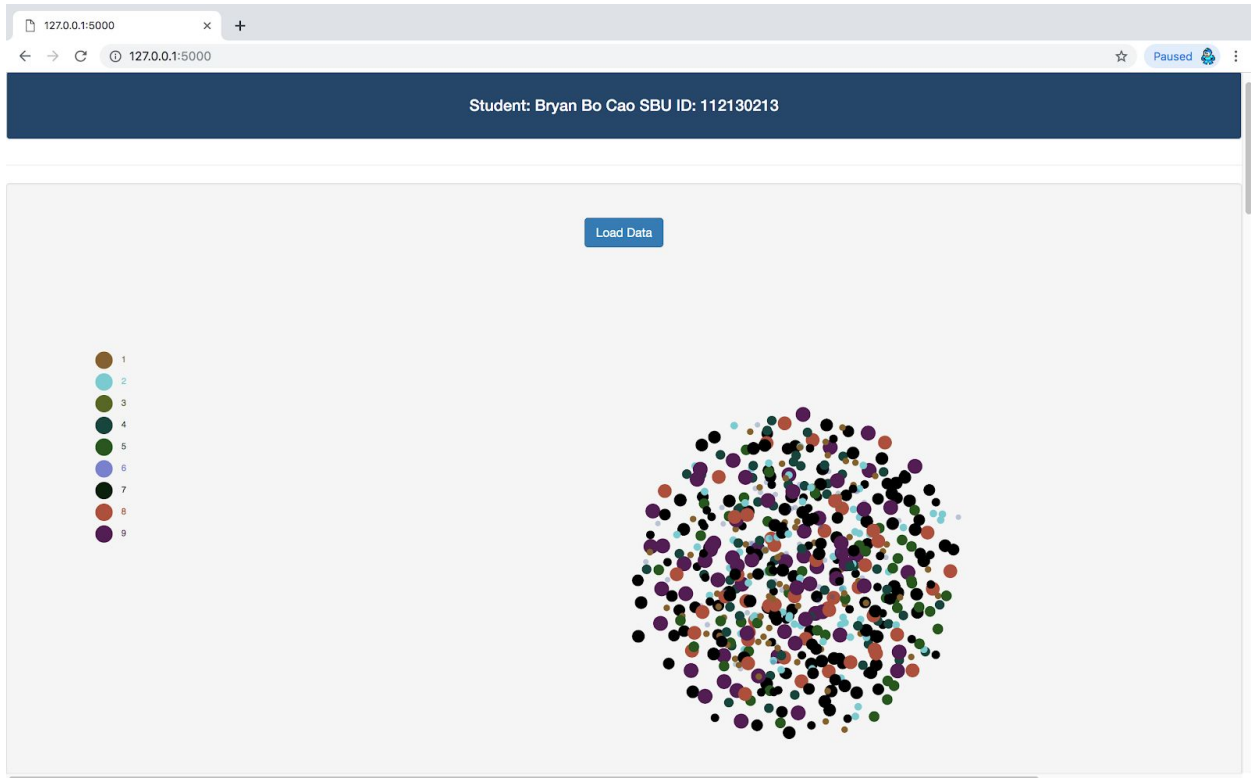
A number of data points are sampled from MNIST, ranging from 300, 600 to 1000, and apply T-SNE on these sampled data. Each node represents a data point from the dataset, labelled with different colors shown in the legends svg at the left side of index.html. Here each data point is a 28x28 image representing a digit. At first all data points look shuffled, and they gradually form clusters with other similar images using euclidean distance. The converged result can be seen in the last image that forms different clusters.

#### 300 Data Points

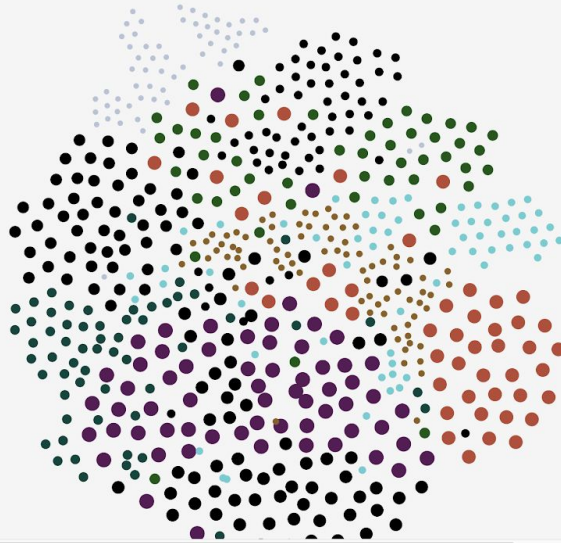




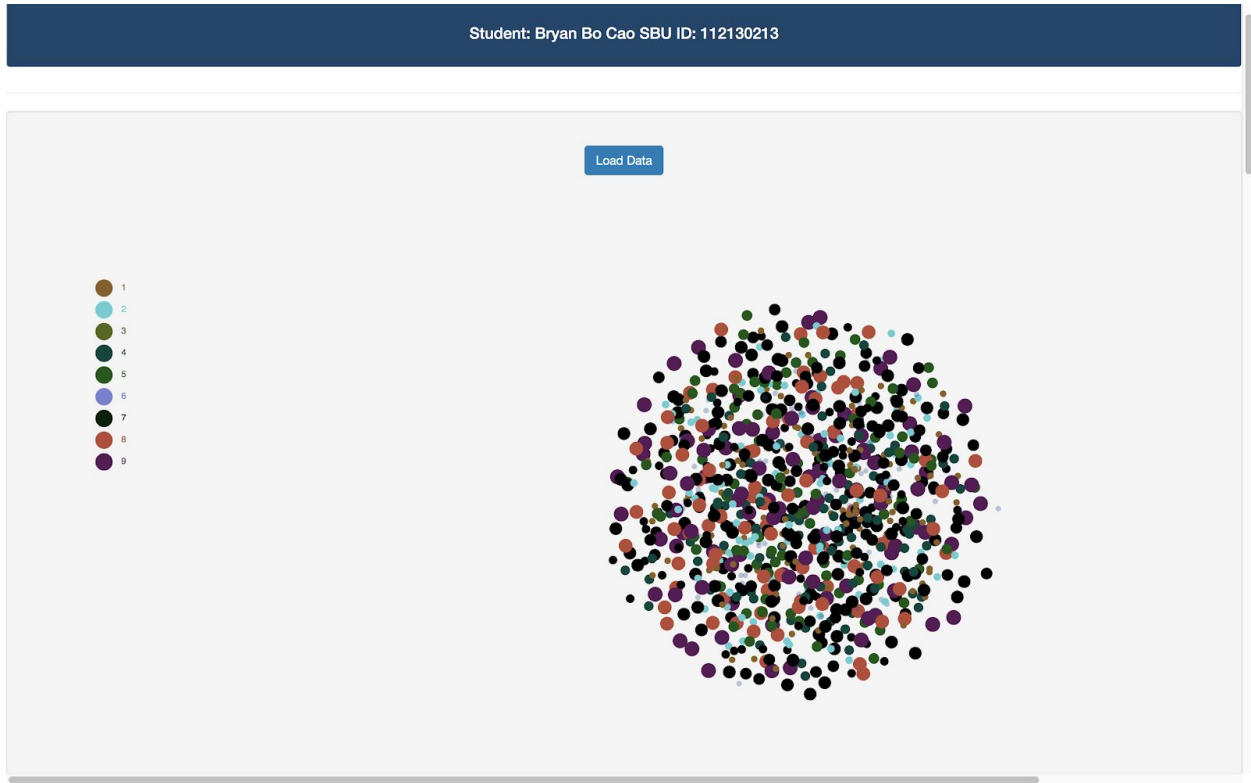
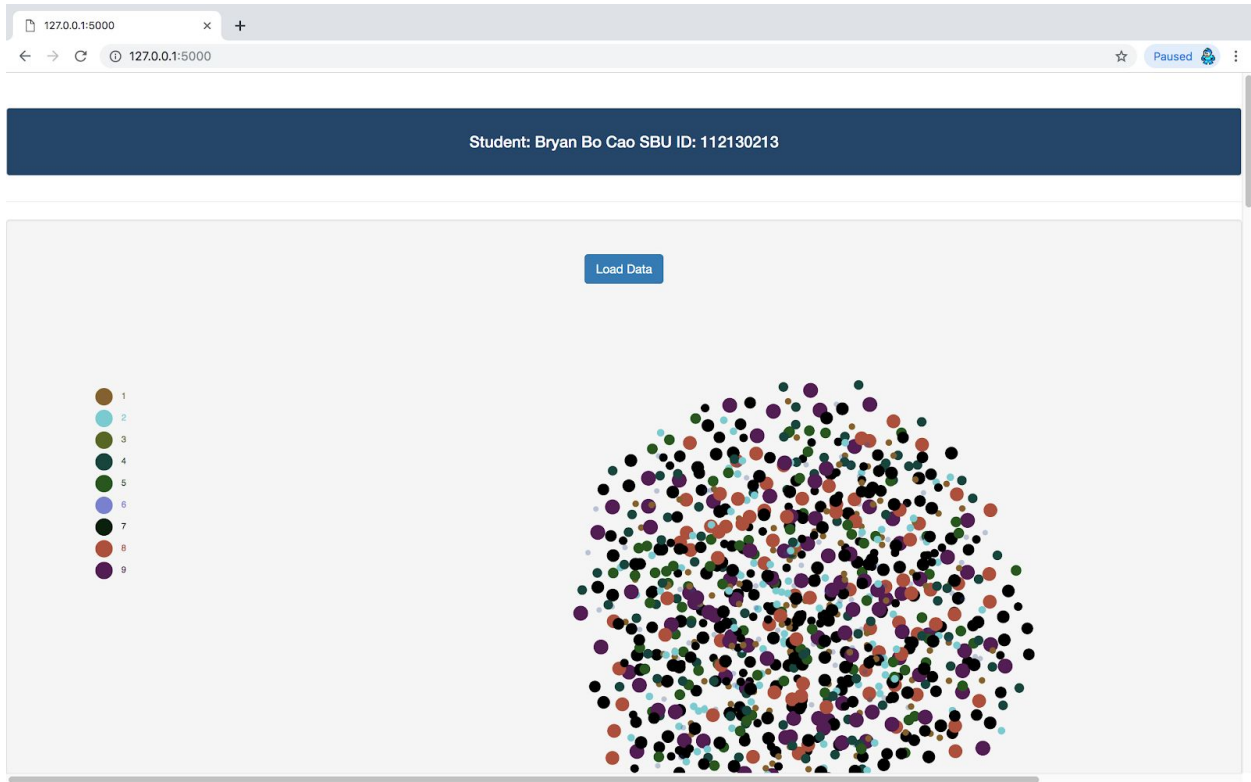
600 Data Points



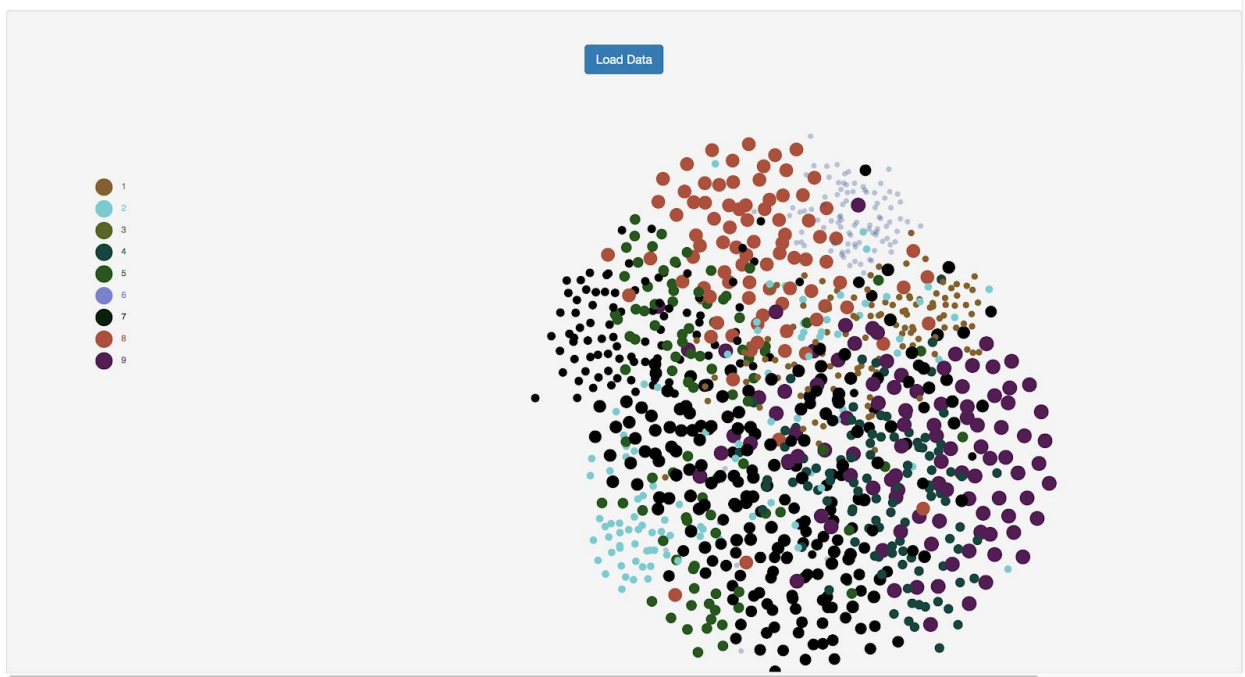
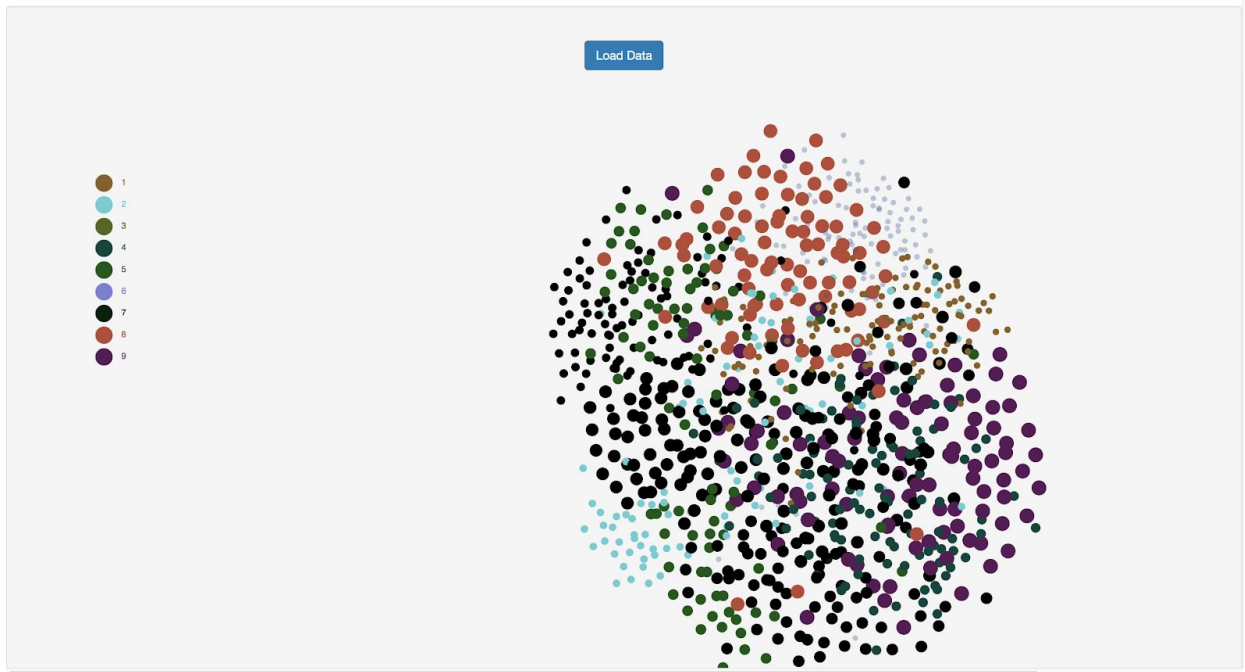
Load Data

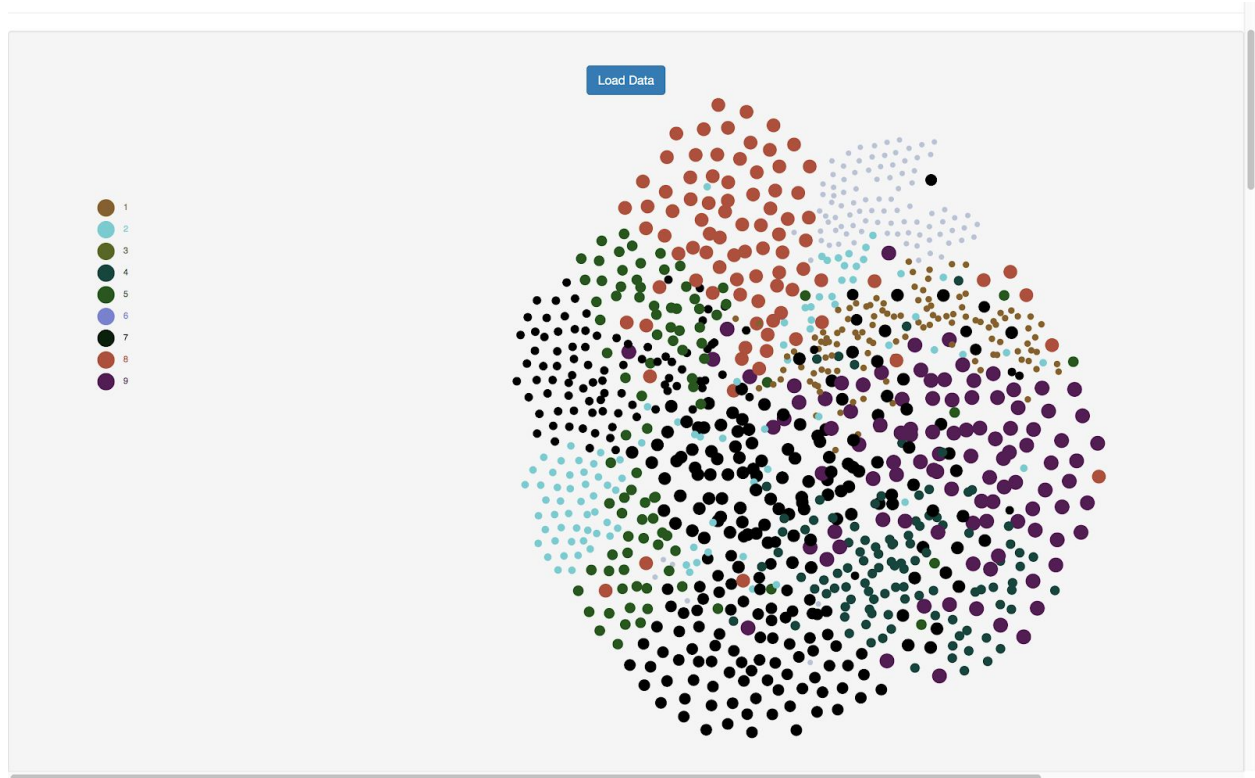


1000 Data Points









### Interesting Findings

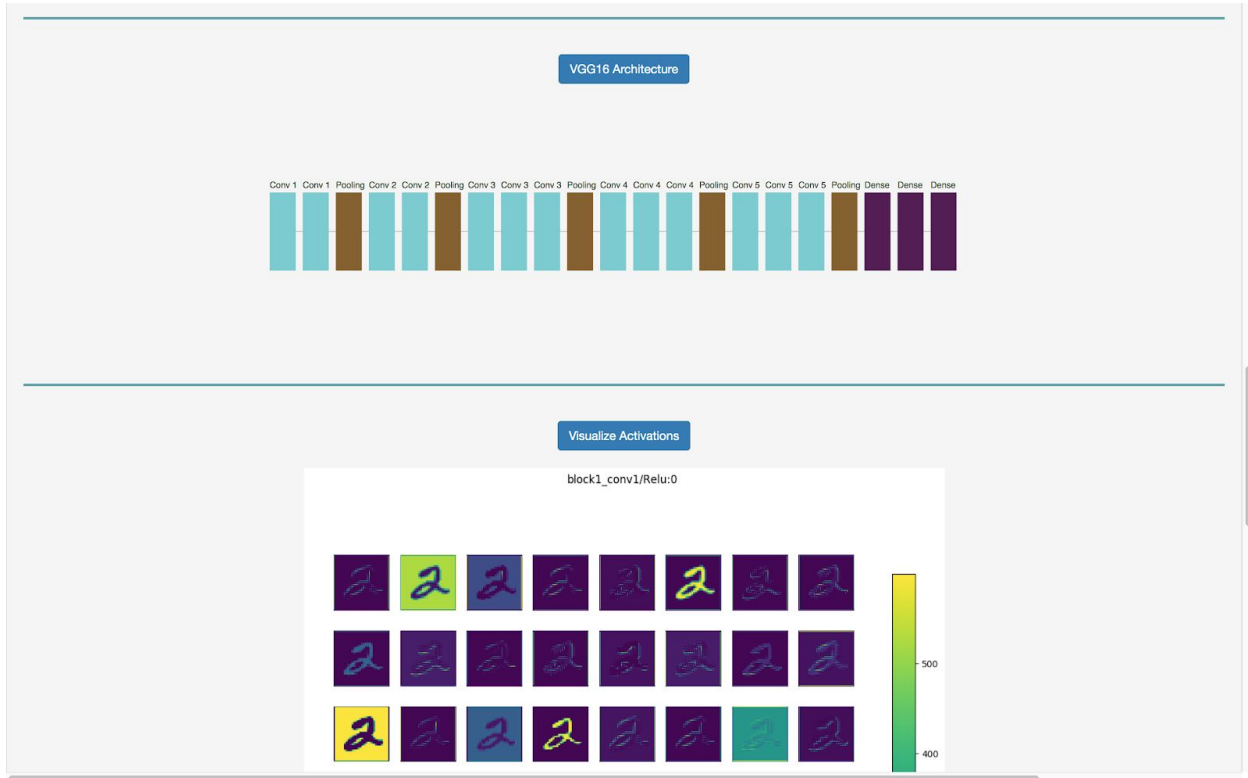
We can find some interesting findings from the above picture. For instance, on the top, the cluster representing 6 and 8 are adjacent to each other, which makes sense to us that from human eye's visual perspective, the shapes between 6 and 8 are similar compared to other digits.

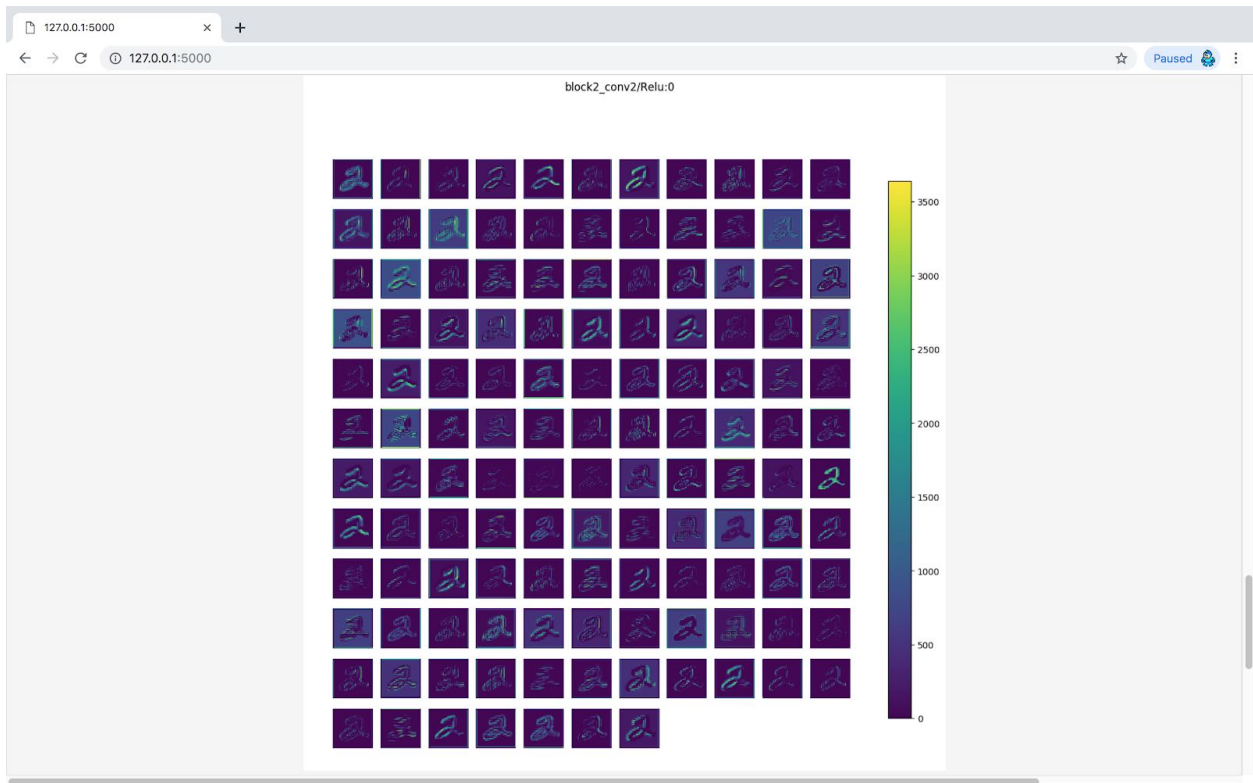
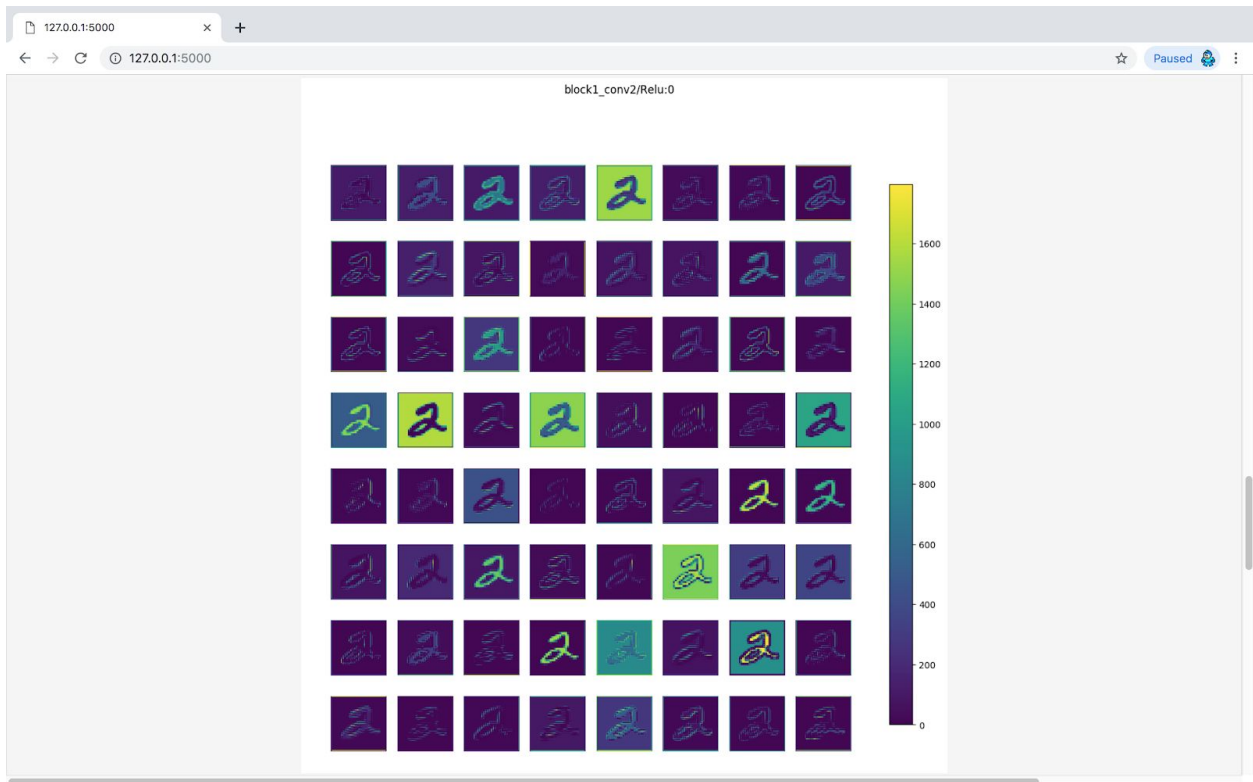
In addition, 6 and 7 look very different, in the visualization above, the corresponding two clusters (top right and bottom) are far away from each other.

## VGG-16 Architecture and Activations

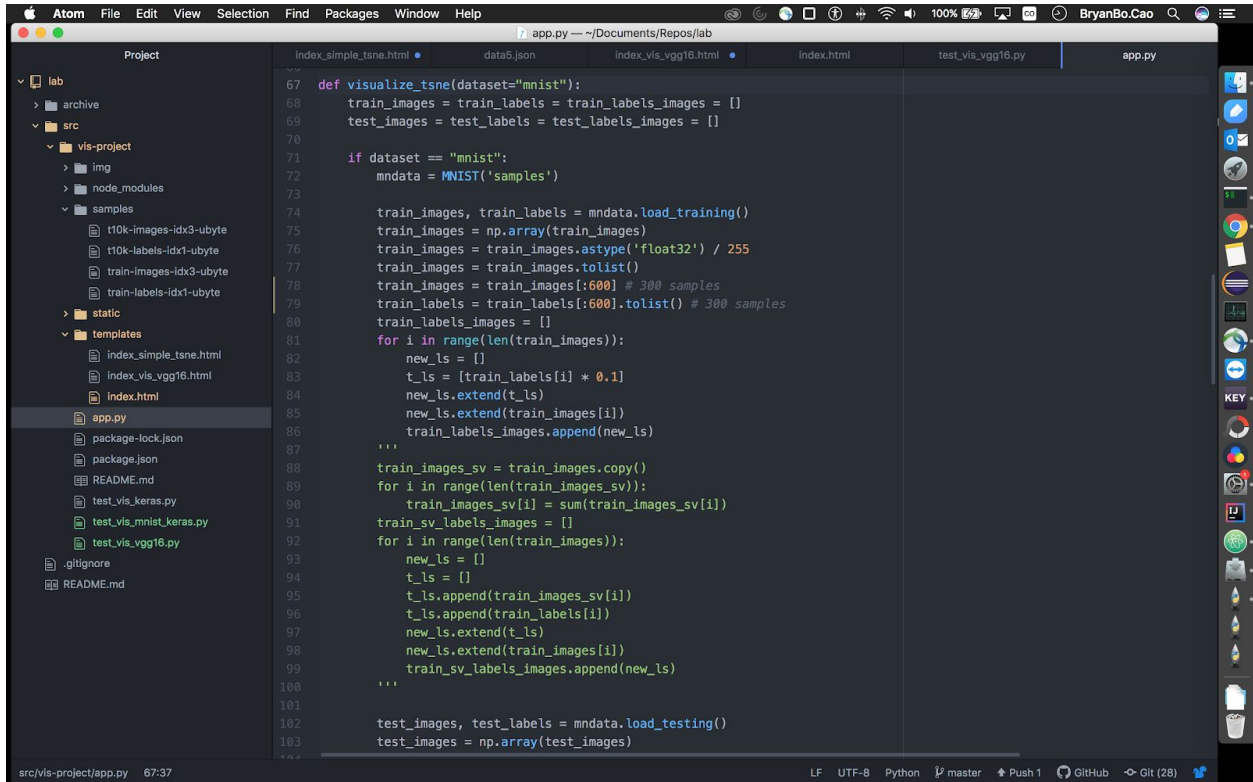
The overall architecture of VGG-16 is visualized in the next svg, followed by the activations from each layer. A image representing “2” is fed into the network. Note that I changed the visualization implementation from Resnet50 to VGG-16.

Previously, the activations usually are displayed by the functions provided from matplotlib in Python. But in my code, these activations are actually displayed in the index.html together with the visualization shown previously in the frontend.





## Snippet of the Code



```
67 def visualize_tsne(dataset="mnist"):
68     train_images = train_labels = train_labels_images = []
69     test_images = test_labels = test_labels_images = []
70
71     if dataset == "mnist":
72         mndata = MNIST('samples')
73
74         train_images, train_labels = mndata.load_training()
75         train_images = np.array(train_images)
76         train_images = train_images.astype('float32') / 255
77         train_images = train_images.tolist()
78         train_images = train_images[:600] # 300 samples
79         train_labels = train_labels[:600].tolist() # 300 samples
80         train_labels_images = []
81         for i in range(len(train_images)):
82             new_ls = []
83             t_ls = [train_labels[i] * 0.1]
84             new_ls.extend(t_ls)
85             new_ls.extend(train_images[i])
86             train_labels_images.append(new_ls)
87         ...
88         train_images_sv = train_images.copy()
89         for i in range(len(train_images_sv)):
90             train_images_sv[i] = sum(train_images_sv[i])
91         train_sv_labels_images = []
92         for i in range(len(train_images)):
93             new_ls = []
94             t_ls = []
95             t_ls.append(train_images_sv[i])
96             t_ls.append(train_labels[i])
97             new_ls.extend(t_ls)
98             new_ls.extend(train_images[i])
99             train_sv_labels_images.append(new_ls)
100         ...
101
102     test_images, test_labels = mndata.load_testing()
103     test_images = np.array(test_images)
```

## Reference

- [1] CVPR18: Tutorial: Part 1: Interpreting and Explaining Deep Models in Computer Vision <https://www.youtube.com/watch?v=LtbM2phNI7I>
- [2] CVPR18: Tutorial: Part 2: Interpreting and Explaining Deep Models in Computer Vision <https://www.youtube.com/watch?v=eZBapwRFsI0>
- [3] [https://www.tensorflow.org/guide/summaries\\_and\\_tensorboard](https://www.tensorflow.org/guide/summaries_and_tensorboard)
- [4] <https://playground.tensorflow.org/>
- [5] <https://github.com/tensorspace-team/tensorspace>
- [6] Harley, Adam W. "An interactive node-link visualization of convolutional neural networks." In International Symposium on Visual Computing, pp. 867-877. Springer, Cham, 2015.
- [7] <http://scs.ryerson.ca/~aharley/vis/conv/>
- [8] <https://www.youtube.com/watch?v=3JQ3hYko51Y>
- [9] <https://www.tensorflow.org/js>
- [10] <https://github.com/szagoruyko/pytorchviz>
- [11] <https://github.com/PaddlePaddle/VisualDL>
- [12] <https://raghakot.github.io/keras-vis/>
- [13] <https://github.com/raghakot/keras-vis>
- [14] <https://github.com/lutzroeder/netron>
- [15] <https://github.com/albermax/innvestigate>
- [16] <http://www.heatmapping.org/>
- [17] Deep Visualization Toolbox <https://www.youtube.com/watch?v=AgkflQ4IGaM>
- [18] Visualizing and Understanding Deep Neural Networks <https://www.youtube.com/watch?v=ghEmQSxT6tw>
- [19] Visualizing and Understanding [http://cs231n.stanford.edu/slides/2017/cs231n\\_2017\\_lecture12.pdf](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture12.pdf)
- [20] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. and Lipson, H., 2015. Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.
- [21] <http://yann.lecun.com/exdb/mnist/>
- [22] Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R. and Samek, W., 2016, September. Layer-wise relevance propagation for neural networks with local renormalization layers. In International Conference on Artificial Neural Networks (pp. 63-71). Springer, Cham.
- [23] <https://lrpserver.hhi.fraunhofer.de/handwriting-classification>
- [24] Zintgraf, L.M., Cohen, T.S., Adel, T. and Welling, M., 2017. Visualizing deep neural network decisions: Prediction difference analysis. arXiv preprint arXiv:1702.04595.
- [25] Samek, W., Wiegand, T. and Müller, K.R., 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296.

Note that in @257 in Piazza, submitting the report by 11pm May 1st would be fine, so I added more results here before the deadline.

CSE 5648

Q & AResourcesStatistics

Search Companies

2

Bo Cao

examlogisticsother

Question History:

? question

121 views

3:22AM

preliminary report submission deadline

Hi Professor,  
For project proposal you had mentioned that we can submit our proposals by next day of deadline till 11pm without any penalty.  
Can you please confirm if this is applicable for preliminary report submission too? If we submit by 1st May 11 pm is that okay?  
Thanks

project

editundo good question2Updated 1 day ago by Anonymous

S the students' answer, where students collectively construct a single answer

Click to start off the wiki answer

I the instructors' answer, where instructors collectively construct a single answer

applies to any of the deadlines

undo thanks5Updated 1 day ago by Klaus Mueller