Final Project Report

CS 6995 - Advanced Data Visualization

Michael Young

Team members

I'm doing this thing solo. UID: u0742759.

Viewing the project

The project is served via GitHub Pages here: https://mkcyoung.github.io/ML-vis/

Introduction

My final project aims to be an interactive exploration of neural networks using the MNIST handwritten digit dataset as the medium through which this exploration takes place. Users can:

1. select different model types to explore
2. build, train, and evaluate the performance of the model
3. peer into the inner workings of the network by visualizing what's happening with the first layer nodes
4. draw their own digits and get real-time feedback of the model's predictions

All of these features combine to give the user a lightweight, interactive and exploratory introduction to neural nets that will hopefully spur on an ongoing interest in the topic.

My primary motivations for choosing this particular project are twofold:

1. Because I don't have much background in machine learning, I wanted to dive into the topic and develop some familiarity with a machine learning library (in this case, it happened to be tensorflow.js), and
2. it resides in the perfect space between challenge and feasibility with respect to my d3/web-vis skills.

Related works and sources

The spirit and flavor of this tool were inspired by many examples we saw in class [1, 2 ,3, 4]. The tool itself was developed using tensorflow.js and much of the code was adapted from this tutorial. Additionally, I found this article immensely helpful as a guiding star in constructing my vis and working with the tfvis.js library. Some other sources I found helpful for specific tasks:

- Working with canvas
- Updating training metrics in real-time
- Visualizing activation maps
- Understanding neural nets and machine learning in general: 1 , 2
- And of course the tensorflow.js and tfvis-js APIs were wonderfully helpful.

Project Objective

My primary objective for this project is for it to be a compelling and engaging exploratory and educational tool for anyone with even a passing interest in neural networks. I intended for it to be simple enough that anyone could get an idea of what's happening yet complex enough that someone with more advanced knowledge of the topic would still find it at least somewhat interesting. Again, my hope is that someone using this tool without background knowledge will think "hey, this is kind of cool, I want to learn more."

Data

As mentioned in the introduction, the data I'm working with is the classic MNIST handwritten digit data set. The developers at google were kind enough to encode this dataset as a ~10MB sprite file and provide some javascript to help load and pipe the data to the model (https://codelabs.developers.google.com/codelabs/tfjs-training-classfication/index.html?index=..%2F..index#2). There wasn't much in the way of insight to glean from the data, but the data did allow for the user to glean some insights into the workings of neural nets, which I'll get to later.

Background

Neural nets have been around for decades (and if we're talking about neural-neural nets – millennia), but they've recently come into vogue because of their success with a number of relevant problems in computer science and beyond. The extreme usefulness of neural nets, as opposed to other machine learning methods, essentially boils down to their ability to learn complex, non-linear features, without the user having to explicitly define these features manually.

One common issue with neural nets however, is interpretability. With increasing layers and many nodes in each layer, nets can quickly balloon into obscurity when one wants to understand why the model is making certain decisions over others. Because of this, models are often treated as "black boxes." Some cutting edge examples designed to tackle this issue of interpretability of deep neural nets (which we've seen in class) are the activation atlas and summit. Both of these methods attempt to visualize and cluster nodes within the network in order to provide the user with some clarity about what's happening in the model.

My project attempts to capture some of the magic of these techniques, but obviously on a much smaller scale. I'm only visualizing the first layer of weights of my network in an attempt to provide the user with a more intuitive knowledge of how networks work, but not overwhelm them with the entire picture.

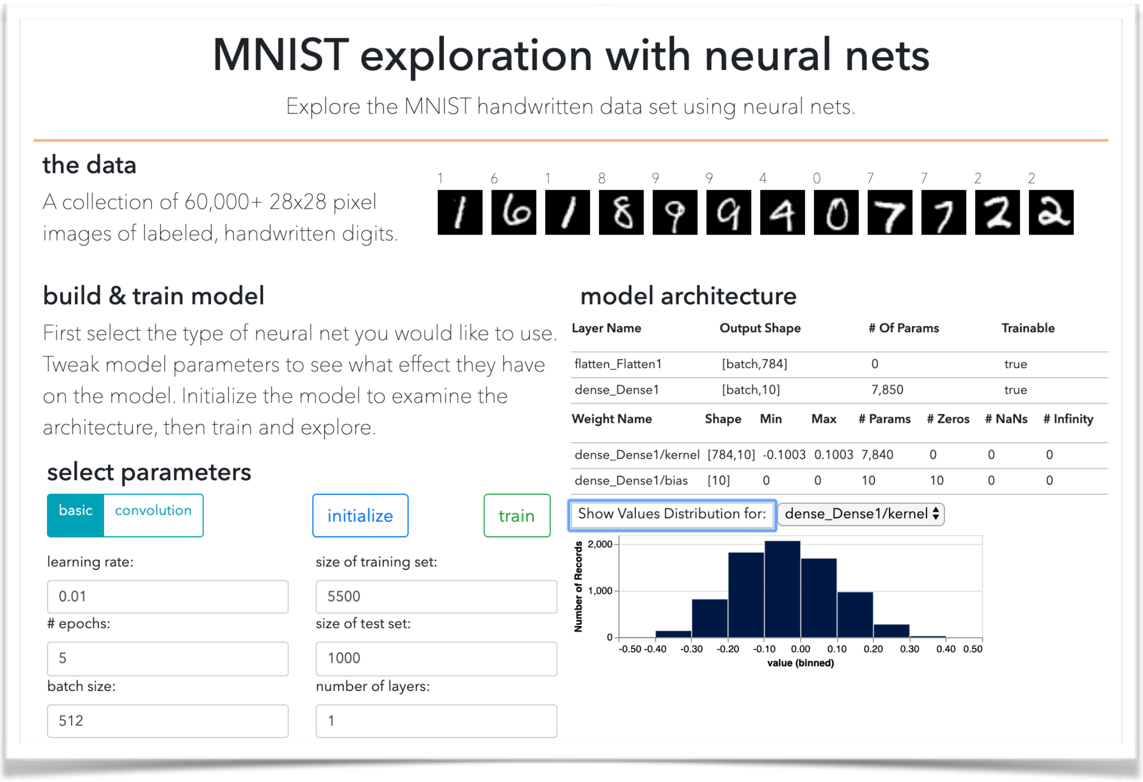
Technical Contributions

If my project has any sort of technical contribution it would be the availability of a user-friendly educational tool to help people get a more intuitive grasp of neural networks. Beyond that I'm not adding anything much to the ML-vis zeitgeist. There are many, many projects out there that have visualized neural nets and done a much more thorough job than what I've accomplished here.

Methods, Results, & Outcomes

This tool consists of three main parts:

1. Building the model



MNIST exploration with neural nets

Explore the MNIST handwritten data set using neural nets.

the data

A collection of 60,000+ 28x28 pixel images of labeled, handwritten digits.

build & train model

First select the type of neural net you would like to use. Tweak model parameters to see what effect they have on the model. Initialize the model to examine the architecture, then train and explore.

select parameters

learning rate: 0.01
# epochs: 5
batch size: 512
size of training set: 5500
size of test set: 1000
number of layers: 1

model architecture

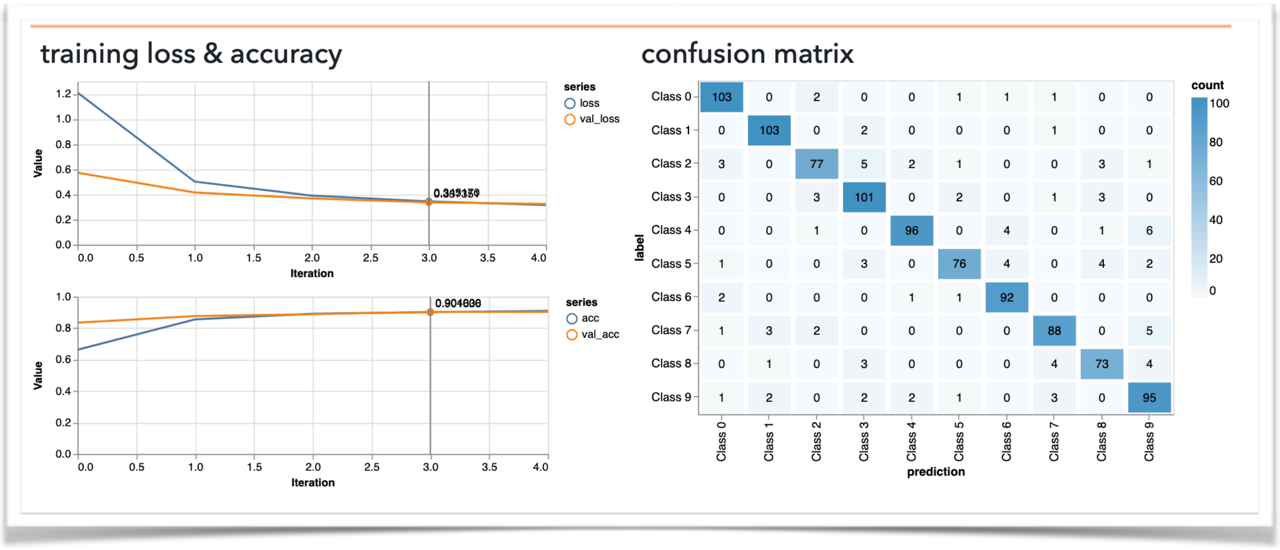
Layer Name	Output Shape	# Of Params	Trainable
flatten_Flatten1	[batch,784]	0	true
dense_Dense1	[batch,10]	7,850	true

Weight Name	Shape	Min	Max	# Params	# Zeros	# NaNs	# Infinity
dense_Dense1/kernel	[784,10]	-0.1003	0.1003	7,840	0	0	0
dense_Dense1/bias	[10]	0	0	10	10	0	0

Show Values Distribution for: dense_Dense1/kernel

At the top of the vis, the user is shown 12 random samples of the dataset along with their corresponding labels. Below this, this user is prompted to set up the model. The user first decides if they want to use a standard artificial neural network, or a convolutional neural net. Next they can adjust the learning rate, epochs, batch size, size of training and test sets, and the number of layers. After the user is satisfied with the model, they hit 'initialize' to examine the model architecture. This architecture is generated using tfvis.show.modelSummary.

2. ***Training the model***

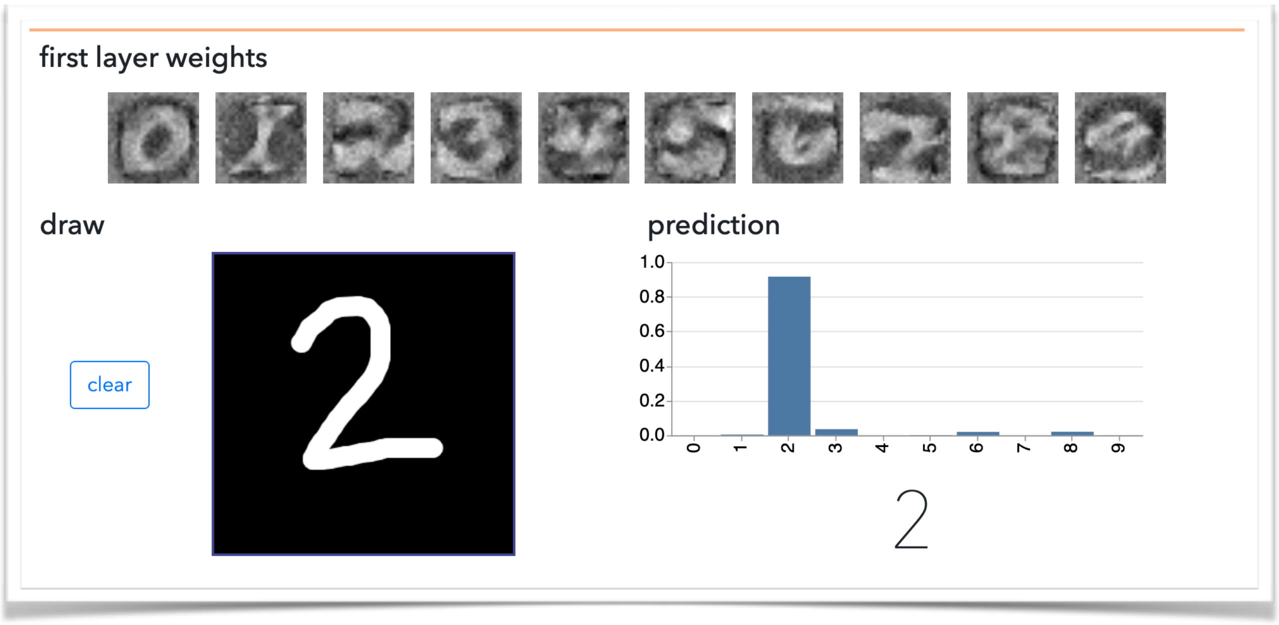


label	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Class 0	103	0	2	0	0	1	1	1	0	0
Class 1	0	103	0	2	0	0	0	1	0	0
Class 2	3	0	77	5	2	1	0	0	3	1
Class 3	0	0	3	101	0	2	0	1	3	0
Class 4	0	0	1	0	96	0	4	0	1	6
Class 5	1	0	0	3	0	76	4	0	4	2
Class 6	2	0	0	0	1	1	92	0	0	0
Class 7	1	3	2	0	0	0	0	88	0	5
Class 8	0	1	0	3	0	0	0	4	73	4
Class 9	1	2	0	2	2	1	0	3	0	95

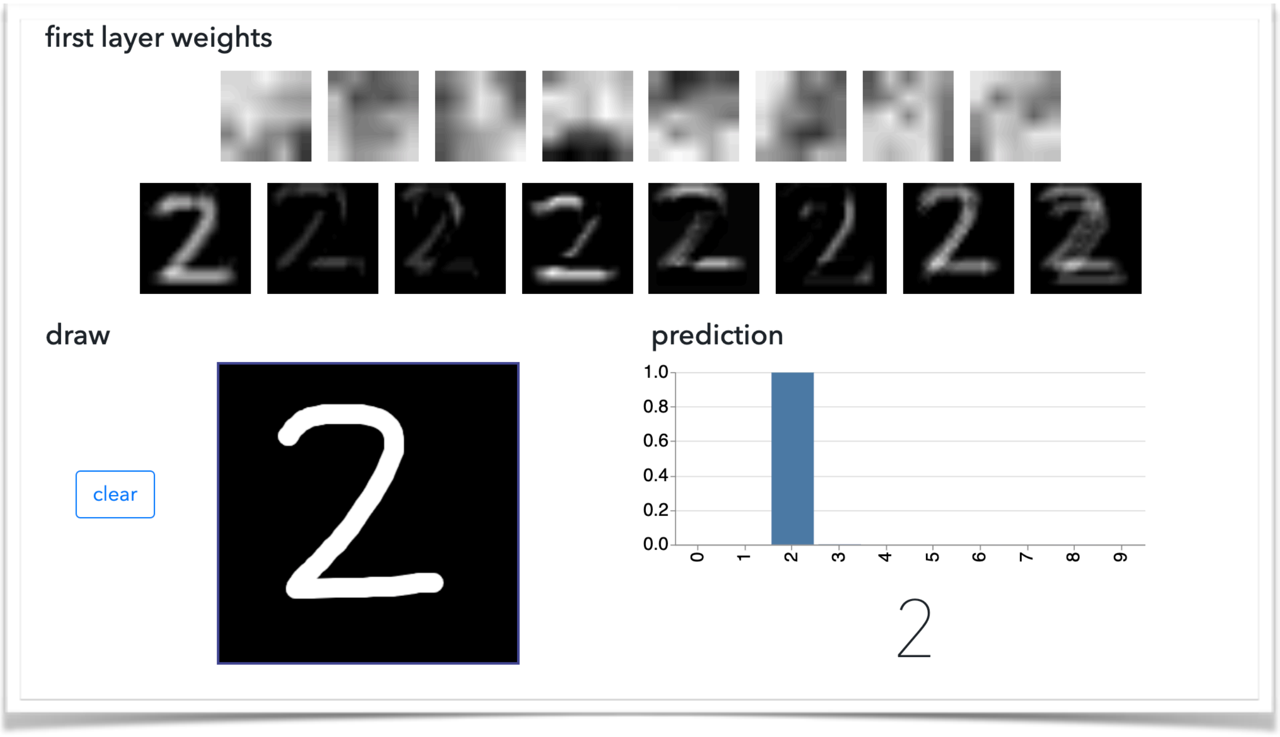
After the user is satisfied with their model set-up, they select "train." After train is selected, plots of the training loss and accuracy (along with the validation loss and accuracy) populate in addition to a confusion matrix of the validation data. These plots are updated at the end of each training epoch using the 'callbacks' property of model.fit along with the tfvis functions 'tfvis.metrics.confusionMatrix' and 'tfvis.show.history.' Here the user can evaluate the model to see if there is any overfitting happening or if the model has trouble distinguishing certain classes from one another. In addition to these plots, a visual representation of the first layer nodes updates at the end of each epoch, allowing the user to see how the model learns over time.

3. ***Exploring the model***

Once the model is finished training, the user has the option to draw their own digits in the browser and receive real-time feedback of the model's predictions. Additionally, if a convolutional net was used for the model, the user is able to see the activation maps generated by the learned convolutional filters. This was accomplished by creating a second model from the first few layers of the original model, and using the output of that model to generate the images. The last layer of the models uses the



If "basic" is selected



If "convolution" is selected

softmax activation, and any additional hidden layers use either sigmoid or relu depending on if basic or convolution is selected. The user can clear their current drawing at any time to draw something new and see what the model predicts.

Here are a few interesting ways to obtain insights from this last exploratory component of this vis:

- adjust the number of layers and see how that changes what the first layer nodes look like (for instance, how do the weights change when there is only one layer vs two?)
- See if you can make any "adversarial" examples based on what the weights of each class are.
- Think about what kind of features each filter captures and how that is apparent in their corresponding activation maps.

Deliverables

As stated at the top of the document,
The project is deployed here:https://mkcyoung.github.io/ML-vis/
And the source code is available here: https://github.com/mkcyoung/ML-vis
Additionally, there is a short video turned in with the report on canvas showing the vis in action.

Evaluation

Here are the metrics I originally defined for success in my proposal along with how well I think I achieved them in the final product:

1) ***Usability*****:** essentially does the thing work, and is it easy to interact with
   Success! Everything works and I think the vis as a whole is fairly straightforward to use.
2) ***Educational value*****:** does this vis offer any sort of educational value to someone learning about ML for the fi rst time or for someone looking to get a refresher
   Here I think my success is mixed. My original idea leaned much more on the educational side, while the final product is a more exploratory thing. I also think the lack of a guided tutorial showing some specific insights ultimately inhibits the overall educational experience because as is, a user would probably need to have some basic knowledge about neural nets to start picking out interesting things. Ultimately I think this tool could be used as an educational supplement but is not sufficient in and of itself to give someone without knowledge any real educational breakthroughs.
3) ***Aesthetics:*** Is the vis pretty?

Eh, here too I think I could have done better, but it's not a complete eye sore (at least I hope not).

4) ***Creativity of views + interface:*** Are the views and controls presented interestingly and do they effectively provide the intended insights?

   I think by this metric I did pretty well. Everything is presented in a very accessible way and I think the interactive components are interesting and exciting.

Overall, I'm happy with the way the vis turned out, and excited to move forward with adding to it / improving it.

Software

I used tensorflow.js + tfjs-vis to implement this project. Additionally, I used HTML, CSS, Javascript, and D3.

Project Summary

My final project aims to be an interactive exploration of neural networks using the MNIST handwritten digit dataset as the medium through which this exploration takes place. Users can:

1. select different model types to explore
2. build, train, and evaluate the performance of the model
3. peer into the inner workings of the network by visualizing what's happening with the first layer nodes
4. draw their own digits and get real-time feedback of the model's predictions

All of these features combine to give the user a lightweight, interactive and exploratory introduction to neural nets that will hopefully spur on an ongoing interest in the topic. Future directions for this work will involve expanding the network visualization aspect to include all layers, as well as adding a guided tour for novice users.