An interactive web tool to visualize ML performance on images

Arunav Saikia[1] arsaikia@iu.edu
Kenneth D Jones[2] kendjone@iu.edu
Franklin Thomas[1] frathom@iu.edu
Vijayalaxmi B Maigur[1] vbmaigur@iu.edu
Sri Harsha Manjunath[1] srmanj@iu.edu

[1]*M.S. Data Science (Residential), Indiana University - Bloomington*
[2]*M.S. Data Science (Online), Indiana University - Bloomington*

Introduction

In this era of deep learning and big data, it is difficult for machine learning researchers and practitioners to understand what their machine learning model is learning, or to debug and fix incorrect predictions. Researchers spend countless hours trying to find the best set of parameters that helps in their specific task, be it classifying an image into a ship or iceberg or segmenting a scene into its elements for self-driving cars. Due to the complexity of the modern deep learning networks and a dearth of tools to visualize and interpret the machine learning models, most practitioners treat their models as a black box. Modern classifiers are also susceptible to adversarial noise patterns that are invisible to humans but change the model's output.

The goal of this project is two way. First, to create a set of visualizations that enables users to understand more about their machine learning models - how the model is learning, where is it going wrong, why is it preferring certain outcomes over others. Second, to build an interactive tool to visualize results on the frontend instead of having to go back to the command line every time data or network parameter changes.

Related Works

Existing tools that allow machine learning (ML) and deep learning (DL) practitioners to visualize the performance of ML and DL algorithms for different tasks require significant domain expertise. Each of these tools has its specialty. For example - Tensor Flow Playground [1] helps users understand the idea behind neural networks without delving into the hard math. It lets users play with a real neural network running in the browser. It also allows the user to click and tweak parameters to see how the network behavior changes. GAN Lab [2] is another such tool which lets users understand deep generative models using interactive visualizations. Karpathy et al [3] also developed a Javascript Library (ConvNetJS) for training DL models directly in the browser.

Most of these existing tools do not allow the user to upload their data for visualizations. Also, some of these tools have restrictions on the model architecture that can be used. The goal of our tool is to provide an easy to use interactive tool that lets users upload their data and play with in-built algorithms, parameters or even use pre-trained models to visualize the impact of these machine learning models on different image datasets. Such a tool will not only act as a quick go-tool to visualize the results of their ML models but also enable users to no longer treat their DL models as a black box and add a dimension of interpretability. The tool will help users visualize and interpret what/how the model is learning and gain a deeper understanding of why it is making certain predictions.

The Tool

After a quick survey we identified the following features to be integrated as part of our tool -

- Support for popular Computer Vision (CV) datasets - MNIST, CIFAR10, Fashion MNIST
- Upload custom dataset
- Choose from different CV tasks – classification, segmentation*, object detection*, image generation*
- Support for popular architectures like VGG/Resnet with pre-trained weights.
- Tweak hyper-parameters
- Upload saved model
- Select visualization

**For this project, we have focused on image classification. But the tool has been built with forward compatibility to integrate new tasks in the future. A snapshot of the tool can be found in the Appendix. Link to GitHub - https://github.iu.edu/ML2Viz/ml-visualization-web*

A user can play with existing popular datasets or upload their dataset. They can choose the computer vision task of their choice. He/She can also decide to train the model from scratch by tweaking hyperparameters or use a saved model to visualize the results.

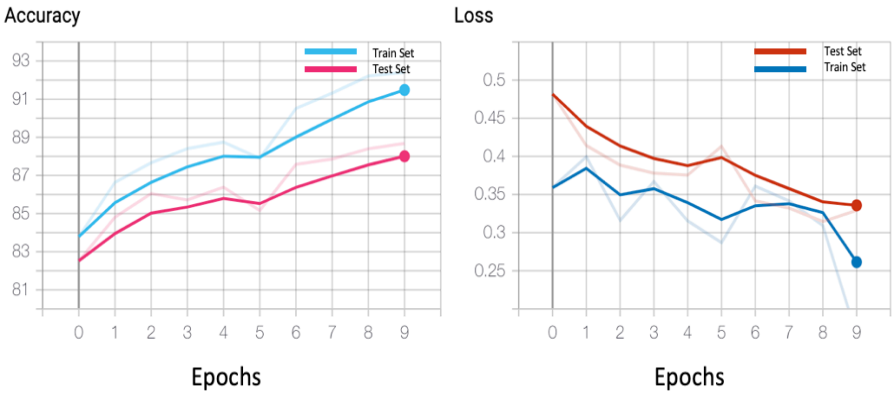
In regard to the tech-stack, the tool uses HTML/CSS for the UI, *PyTorch* [4] for deep learning, *Flask* to handle backend API requests and *TensorBoard* [5] to render the visualizations. The tool also has GPU support and can be easily hosted on a VM like Google Compute Engine running Telsa K80.

Data Analysis and Visualization

We start our analysis with a popular computer vision dataset - Fashion MNIST [6]. The train and test datasets have 60,000 and 10,000 images respectively of 28x28 resolution. The goal of the machine learning model would be to classify each image to one of the 10 labels. Each training and test example is assigned to one of the following labels:

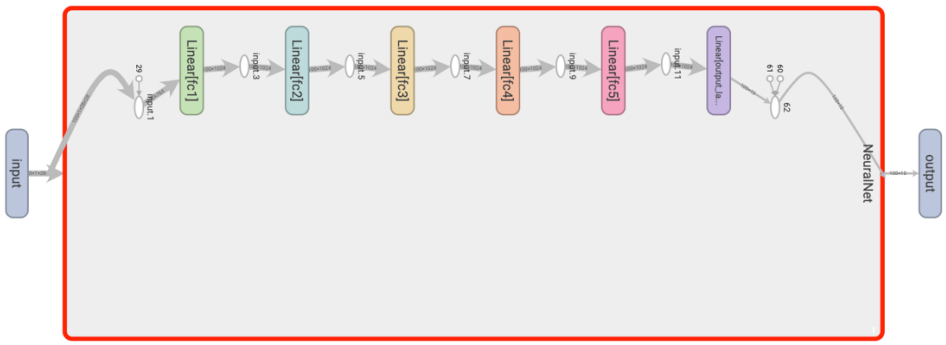
- T-shirt/top
- Trouser
- Pullover
- Dress
- Coat
- Sandal
- Bag
- Ankle boot
- Shirt
- Sneaker

Sample 1:



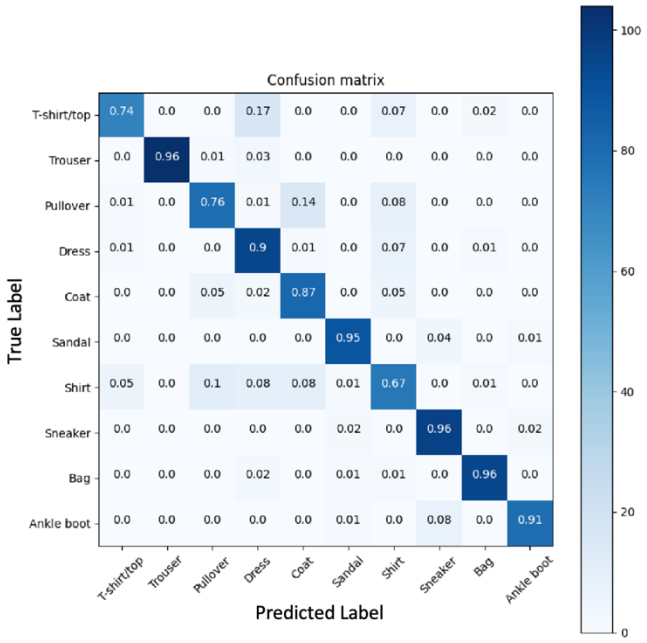
A line graph to show the temporal relationship between machine learning metrics like accuracy and loss (i.e. penalty for bad prediction) over time. Graphic variable hue is used to differentiate between the training and test dataset. Overtime (epoch) we would expect the accuracy metrics to go up and the loss to go down which represents the model is getting better over time. Since the model learns from the train set and predicts on the test set, we can see the metrics are better for the train set than the test set. Also, the test set metrics closely follows the train set indicating the model is not overfitting and will be able to generalize well to unseen data.

Sample 2:



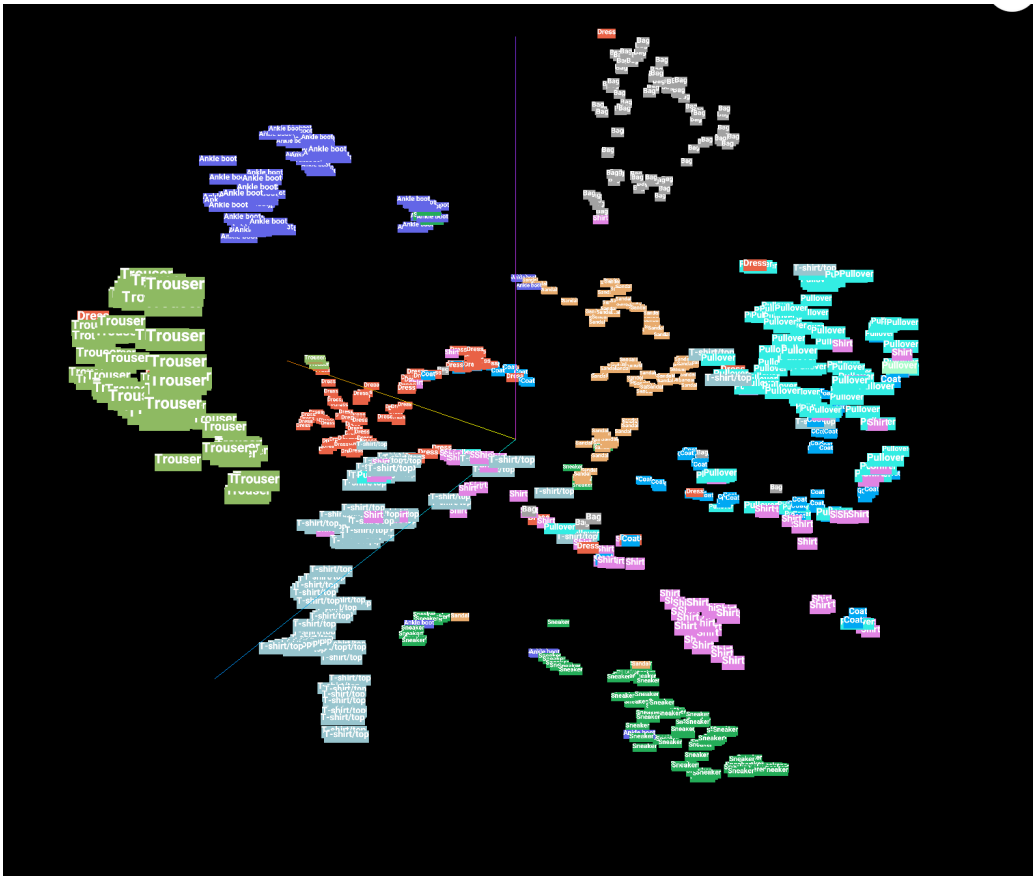
A node-link graph to visualize the neural network architecture. The nodes represent the various layers of the network and directed edges represent how data flows from one layer to the next. Graphic variable hue is used to differentiate between the different layers of the network. This visualization will not only help the user to have a visual representation of their deep learning architectures but also debug any inaccuracies/errors which might crop up in the code while defining the architecture.

Sample 3



A heatmap to show which category of images is difficult to learn. Ideally, we want our classifier to do well with higher values along the diagonals which represent correct classifications. Higher values in the off-diagonals represent misclassifications. From the heatmap above we can see, there is some misclassification between 'T-shirt/top' predicted as 'Dress' and 'Pullover' predicted as 'Coat'.

Sample 4



A 3D projection of data to visualize how the machine learning classifier differentiates between the different classes. Graphic symbol text is used to represent each image, the graphic variable position is used to calculate the location in the 3D plane and graphic variable hue to differentiate between the classes. The 3D coordinate for each image is calculated after applying a dimensionality reduction technique - TSNE [7] to the last feature layer. We can see that some categories are distinct e.g. 'Trouser', 'Ankle boot', 'Bag', 'Sneakers' with well-defined decision boundaries whereas some categories are overlapping or even scattered which leads us to conclude that these categories are more prone to misclassification e.g. 'T-shirt/top', 'Coat', 'Shirt', 'Pullover'

The above visualizations were created with the tool using a 5 layer fully connected architecture running for 10 epochs with a learning rate of 0.001 and SGD optimizer. A user can easily change the dataset or network parameters with the UI and generate their own set of visualizations. If a user uploads a custom dataset, he/she needs to ensure that the folder hierarchy is such that the images are inside sub-directories for different image categories. The tool wrangles through the different sub-directories and creates a train/test set with an 80% and 20% split. The images are resized to 64x64 resolution and pixel intensities for each channel are standardized. The tool can support both RGB and grayscale images.

Validation and Redesign

One of the major issues we faced during this project was to deal with the exorbitant amount of time it takes to train deep learning models. Since training models may take up to hours if not days, depending on the complexity of the model and volume of data, often the user had to wait for a long period for the tool to finish the computation before rendering the visualizations. To overcome this, we have added support in the tool for the user to upload a saved model. The tool then uses the saved model to make the predictions and render the visualizations which save a lot of time.

Another issue we faced was concerning the expressiveness and legibility of the visualizations. Since we used *TensorBoard [5]* which is a library developed by Google to create the visualizations, we were restricted by the library in terms of the amount of information that could be portrayed by each visualization. We could use D3.js to create these visualizations and not rely on the library but then it would be akin to reinventing the wheel. A future development could be to use D3.js to create visualizations related to explainable AI and understanding DL results, which are currently not supported by *TensorBoard [5]*.

Conclusion

The most exciting aspect of this project is the endless possibilities in terms of functionality which could be added to the tool. We could extend the project to create visualizations for different computer vision tasks like object detection, image segmentation etc. We could add support for modern and state of the art architectures. And lastly, we could collaborate with different researchers around the world who are working with interpretable AI to add new and informative visualizations as part of the tool.

Acknowledgements

The authors wish to thank Dr. David J. Crandall and Yingnan Ju for their invaluable suggestions. The authors also wish to thank Katy Borner, Michael Ginda, and Andreas Buckle of CNS center at IU Bloomington for their instructions and support. This work was part of the Information Visualization (ENGR-E 583) course for Spring 2020 at Indiana University, Bloomington.

References

1. Tensorflow playground - Smilkov, Daniel, Shan Carter, D. Sculley, Fernanda B. Viégas, and Martin Wattenberg. "Direct-manipulation visualization of deep networks." *arXiv preprint arXiv:1708.03788* (2017).
2. GAN Lab - Kahng, Minsuk, Nikhil Thorat, Duen Horng Polo Chau, Fernanda B. Viégas, and Martin Wattenberg. "Gan lab: Understanding complex deep generative models using interactive visual experimentation." *IEEE transactions on visualization and computer graphics* 25, no. 1 (2018): 1-11.
3. ConvNetJS - Karpathy, Andrej. "Convnetjs." *URL http://cs. stanford. edu/people/karpathy/convnetjs* (2014).
4. PyTorch - Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen et al. "PyTorch: An imperative style, high-performance deep learning library." In *Advances in Neural Information Processing Systems*, pp. 8024-8035. 2019.
5. TensorBoard - Mané, D. "TensorBoard: TensorFlow's visualization toolkit, 2015."
6. Fashion MNIST - Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747* (2017).
7. TSNE - Maaten, Laurens van der, and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of machine learning research* 9, no. Nov (2008): 2579-2605.

Appendix

Snapshots of the tool – HTML UI and visualizations rendered by Tensorboard

User decides to upload custom dataset and use a saved model by specifying path to saved model

ML2Viz

Choose a dataset: Other (Your Own)
Choose the file: Choose File No file chosen Upload

Please select your task:
- (•) Classification
- ( ) Segmentation (not in use)
- ( ) Object Detection (not in use)

What do you want to do?: Upload trained model

Trained Model Path

Batch Size: eg. 100

Visualizations :

- [x] Image Confusion Matrix [x] 3D Projector Tool
- [x] Network Architecture
- [ ] Important Images (not in use) [ ] Some Other Viz (not in use)

Number of images to visualize eg. 1000

Train and Visualize
Tensorboard
Reset

User decides to use predefined dataset and train own model by tweaking hyperparameters

ML2Viz

Choose a dataset: Fashion MNIST

Please select your task:
- (•) Classification
- ( ) Segmentation (not in use)
- ( ) Object Detection (not in use)

What do you want to do?: Train my own model

Please select model architecture:

- (•) 5 layer FCN
- ( ) VGG16
- ( ) Resnet34 (not in use)

Epochs: eg. 5

Batch Size: eg. 100

Optimizer:

( ) Adam (•) SGD

Learning Rate: eg. 0.01

Visualizations :

- [x] Image Confusion Matrix [x] 3D Projector Tool
- [x] Network Architecture
- [ ] Important Images (not in use) [ ] Some Other Viz (not in use)

Number of images to visualize eg. 1000

Train and Visualize
Tensorboard
Reset