The XAI Discovery Platform

Shane Mueller
Michigan Technological University

Gary Klein
Macrocognition, LLC

Robert R. Hoffman
Institute for Human and Machine Cognition

Bill Ferguson
Raytheon/BBN

This material is approved for public release. Distribution is unlimited. This material is based on research sponsored by the Air Force Research Lab (AFRL) under agreement number FA8650-17-2-7711. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL or the U.S. Government.

Cite as:

Mueller, S.T., Klein, G., Hoffman, R.R. and Ferguson, B.. (2021). "The XAI Discovery Platform." Technical Report, DARPA Explainable AI Program.

Abstract

In this report, we describe the implementation of the XAI Discovery Platform (DP). The discovery platform is a concept that supports users' and developers' exploration ad self-explanation of an image classifier data set, to enable them to identify patterns, see anomalies, and identify predictable rules for using the system and anticipating its weaknesses. We describe several phases of implementing the DP, and describe the different views and browsing modes it enables. We demonstrate its use on two different image classifier systems (the MNIST data set and the FGVC data set), and finally, we describe how to obtain and implement the system for a new image classifier.

Outline

Background 3
Overall Interface Design 3
The Development of the Discovery Platform 8
Prototype I: MNIST Digits using SVM classifier 9
Prototype II: Extending for the FGVC Data Set 10
Insights From the Discovery Platform 13
Implementation Details 14
Limitations and Potential Future Features 17
Acknowledgements 18
References 18

Background

In this Technical Report, we describe the development of the XAI discovery platform (DP).This platform is proposed as a non-algorithmic explanation method that supports self-explanation and discovery about an AI system—specifically targeting AI image classifiers. Notionally, a discovery platform should allow a user or developer to browse, filter, compare, and contrast example images for which an image classification system has been applied. This interactivity will permit the user or developer to see typical patterns of accuracy and error, to identify specific problems, to compare different classes or classifications of the system, and to find outliers and anomalies. Although the notion of a discovery platform is general and could be implemented as part of a general explanation interface, we provide a reference platform implemented within R Shiny that can be used on reasonably large classified image sets (tested on sets up to 10,000 images). In this report, we describe some of the implementation lessons learned by implementing the DP on two distinct AI classification data sets, provide details on the implementation of the system, and identify some of the kinds of discoveries it permits.

We have conducted three iterative rounds of implementation of the DP, each centered on a novel classifier data set. This has led to several refinements and insights to support the system.

Overall Interface Design

The DP has five tabs to help explore and compare different cases, organized to support a global-to-local exploration, which include (1) an overview; (2) category explorer; (3) contrast explorer; (4) case explorer, and (5) table filter.

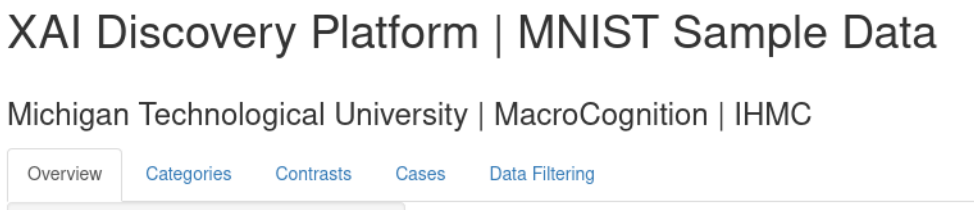

XAI Discovery Platform | MNIST Sample Data
Michigan Technological University | MacroCognition | IHMC
Overview
Categories
Contrasts
Cases
Data Filtering

Figure 1. The main Tabs of the Discovery Platform.

The Overview Tab

The Overview tab provides an overview of the ground truth categories and labels, using several information visualization approaches (see Figure 2). First, it shows an accuracy barplot. This helps identify image categories that are poorly recognized. In Figure 2 (below), it shows that overall accuracy of the system is around 50%, but some characters (like 0) are recognized much more accurately than others (like 8). Next, it shows a heatmap of the confusion matrix, which help identify specific errors that are frequently made. In Figure 2, darker cells represent larger proportions, so it shows that 3 is rarely confused for 4, but 9 is more frequently confused for 4. Finally, it performs a visualization technique called correspondence analysis (CA) using the ca library in R. This puts both ground truth categories and labels in a common 2-dimensional space to visualize which categories are likely to be confused, and if responses for some category are drawn toward another category. In Figure 2 (below), the 7, 9, and 4 appear in a small region of space, indicating relatively many confusions between these classes.

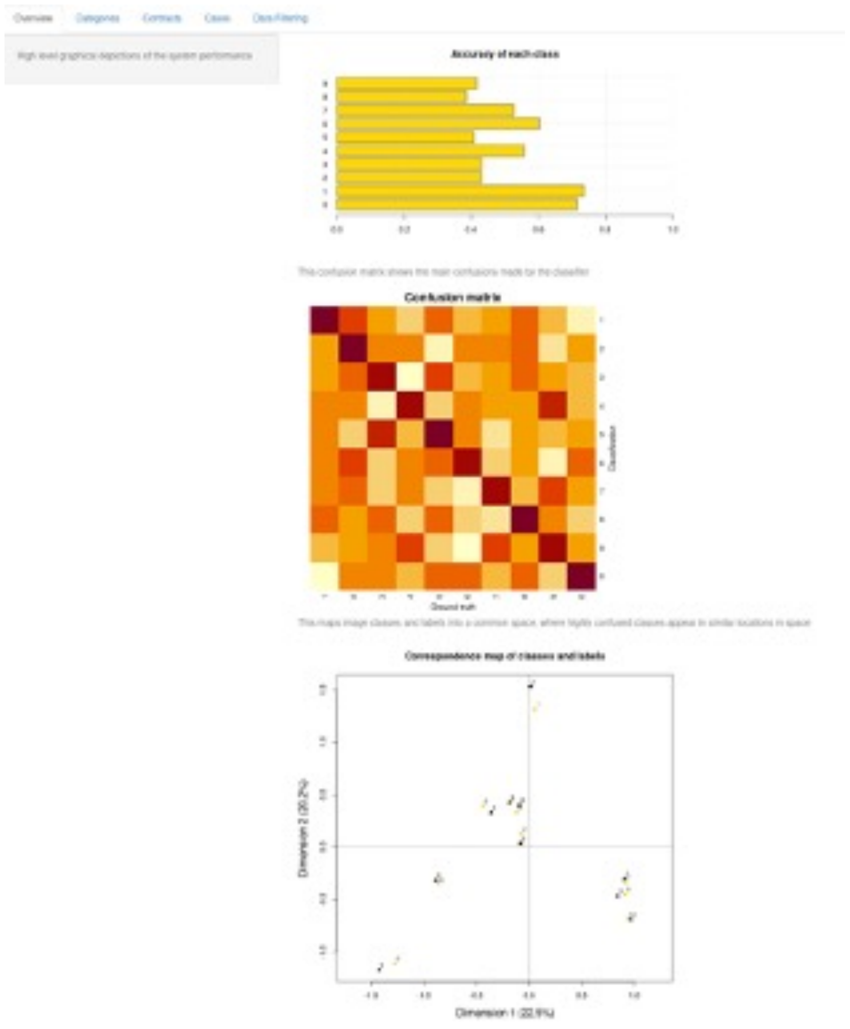
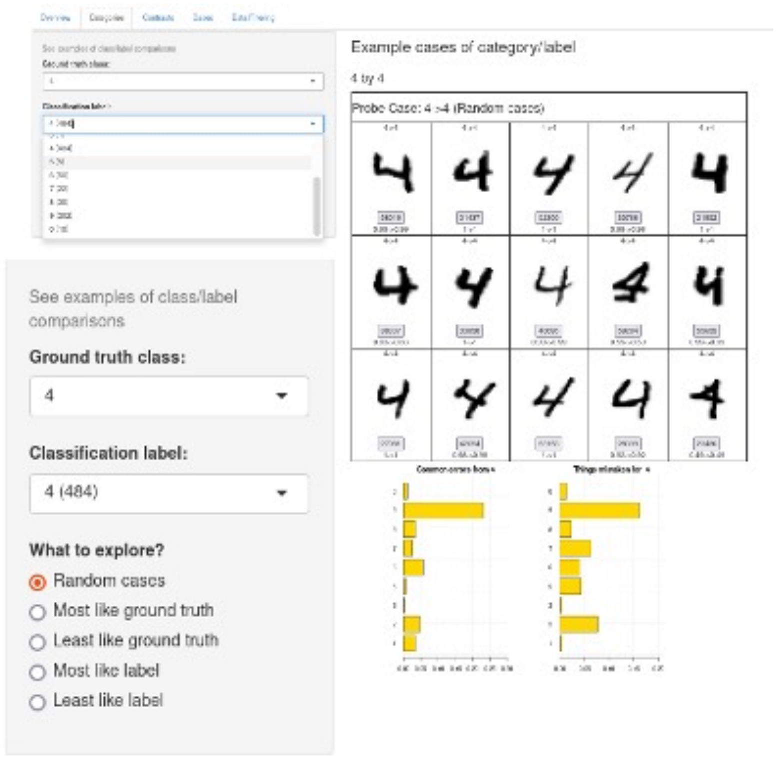


Figure 2. Overview Tab of the Discovery Platform.

The Category Explorer

The Category Explorer tab is intended to allow the user to select a particular category label of the image, and see examples of the kinds of labels made by the AI. Initially, one may wish to examine a random set of responses (by selecting 'any' classification label), but the explorer also allows one to select the particular response. Thus, one can look at correctly identified cases, and then explore particular errors. When selecting a response category, the DP provides a count of how many are in each case, to easily identify the most common errors.

The DP selects random examples that match the selected criteria. However, the examples can be selected based on similarity to specific label categories or examples. These selections can help examine typical and atypical examples. These are selected not based on visual similarity, but based on the similarity of the posterior or activation scores produced by the classifier, using Kullback-Leibler divergence to rank order examples.



Example cases of category/label

4 by 4

Probe Case: 4 ->4 (Random cases)

See examples of class/label comparisons

Ground truth class: 4

Classification label: 4 (484)

What to explore?

[✓ Random cases]
Most like ground truth
Least like ground truth
Most like label
Least like label

Figure 3. Screenshot of the Category Explorer pane of the Discovery Platform. An expanded view of the selection pane is shown in the lower left of the figure.

The Contrast Explorer

The third tab is like the category explorer, but allows the user to specify two ground-truth/label pairs. This is most useful for comparing, for example, cases that were correctly identified with ones in which a particular error was made. We have found this view especially useful for helping suggest rules of thumb that predict errors (e.g., 4s drawn with a triangular top tend to be mistaken for 9s). Like the Category Explorer, selection of cases can be done randomly, or based on similarity to the contrasting case.

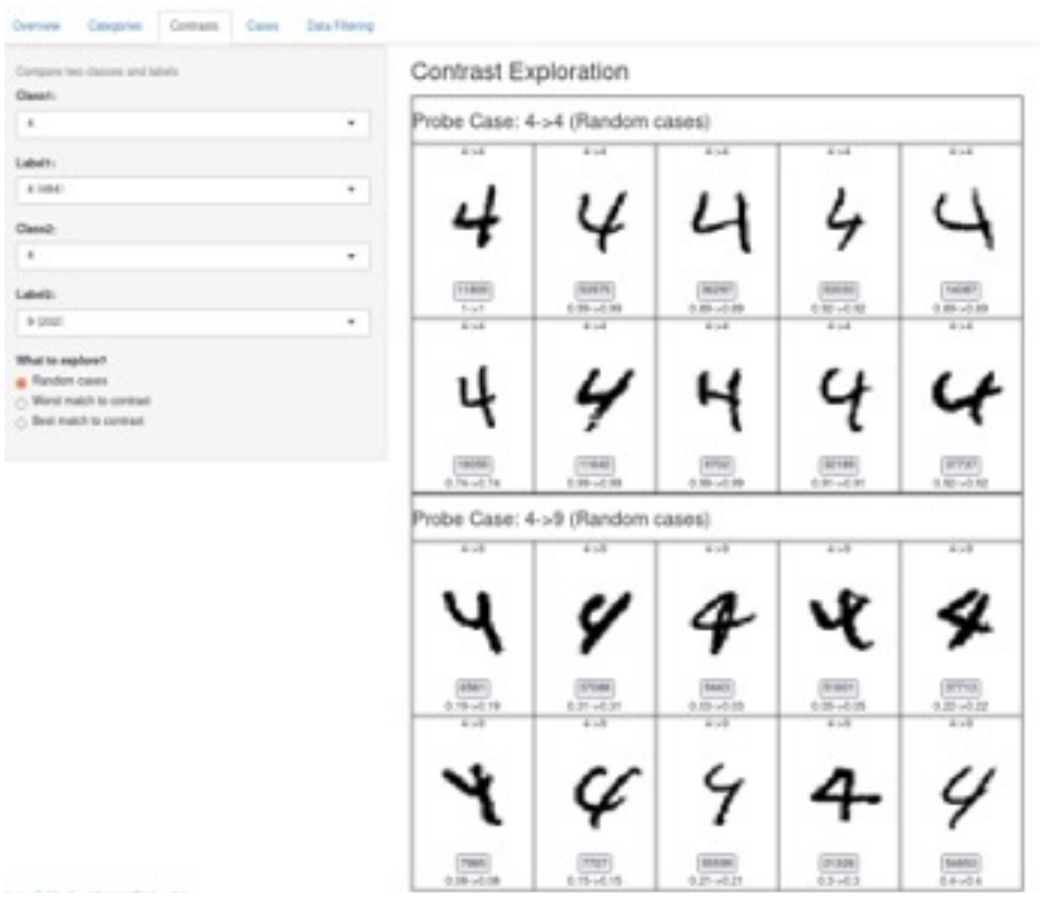


Figure 4. Screenshot of the Contrast Explorer pane of the discovery platform. Example shows correctly identified 4s versus 4s mistaken for 9s by the classifier.

The Case Explorer

Whenever the embedded button for a particular case is clicked on any of the Explorer tabs, the case explorer tab is loaded. This provides detailed information about that case (the ground truth, a distribution of other likely answers, and the best answer), and a table of related cases is also shown. Like the Category and Contrast Explorer tabs, these can be organized by cases that are most like the target, least like the target, or random, to help the user get a better and more representative sampling of cases.

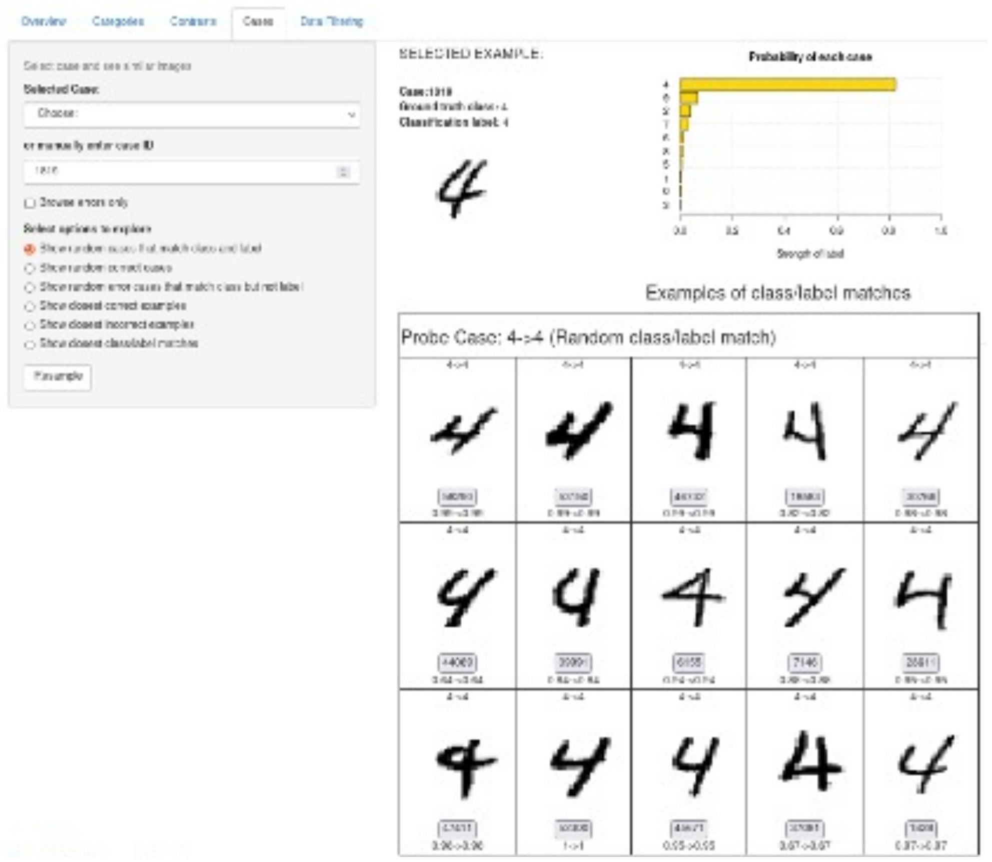
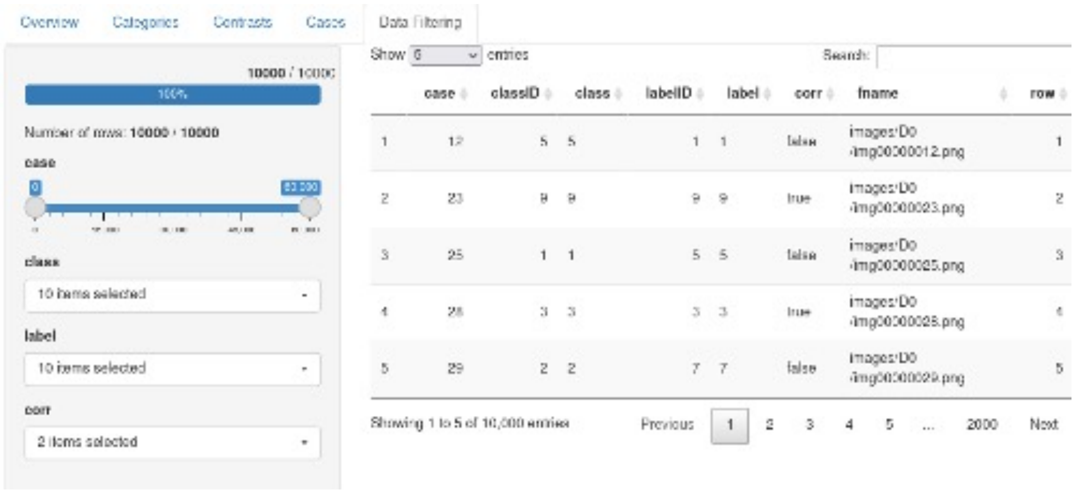

Overview
Categories
Contrasts
Cases
Data Testing
Select case and see similar images
Selected Case:
Choose:
or manually enter case ID
Browse errors only
Select options to explore
Show random cases that match class and label
Show random correct cases
Show random error cases that match class but not label
Show closest correct examples
Show closest incorrect examples
Show closest class/label matches
Resample
SELECTED EXAMPLE:
Case:1919
Ground truth class: 4
Classification label: 4
Probability of each case
Strength of label
Examples of class/label matches
Probe Case: 4->4 (Random class/label match)

Figure 5. The case explorer shows information about a particular case, including a reference number, classifier performance statistics, and similar cases based on various criteria.

The Table Filter

The final tab permits filtering based on ground truth category, label, accuracy (correct/incorrect), and additional data columns specified by the user. This may help the user exclude some cases, and narrow in on other cases. Once its parameters are set, every other tab uses only the cases that match the filtering criterion.



10000 / 10000

Number of rows: 10000 / 10000

case

class: 10 items selected

label: 10 items selected

corr: 2 items selected

Show 5 entries

Search:

	case	classID	class	labelID	label	corr	fname	row
1	12	5	5	1	1	false	images/D0/img00000012.png	1
2	23	9	9	9	9	true	images/D0/img00000023.png	2
3	25	1	1	5	5	false	images/D0/img00000025.png	3
4	28	3	3	3	3	true	images/D0/img00000028.png	4
5	29	2	2	7	7	false	images/D0/img00000029.png	5

Showing 1 to 5 of 10,000 entries

Previous 1 2 3 4 5 … 2000 Next

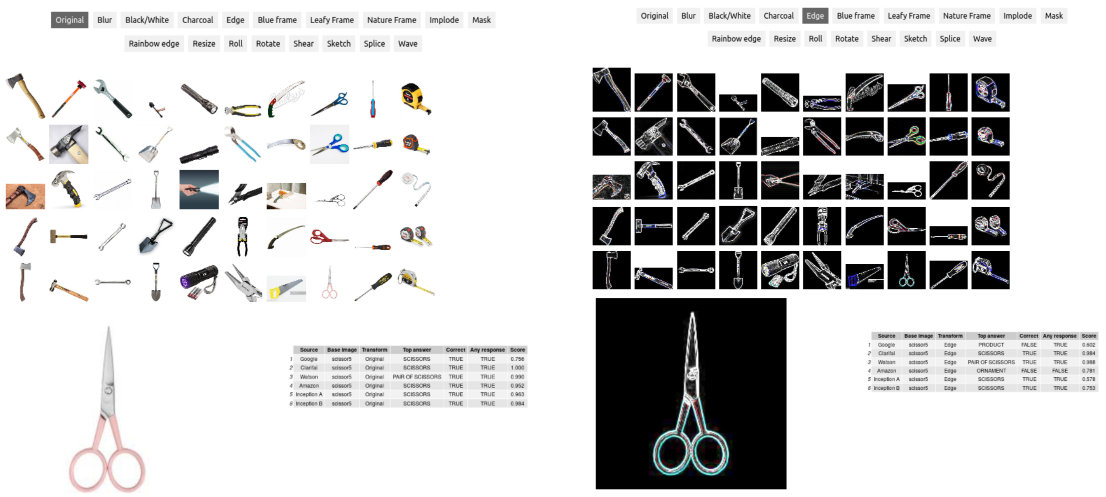
Figure 6. The Table Filter pane allows subsets of cases to be selected based on class, label, accuracy, or other criteria.

The Development of the Discovery Platform

Development of the discovery platform involved several main phases, which we document here.

Initial Concept

Initially, the DP implementation was inspired by two internal systems: one was a browser for an image classification data set used to examine how different image transformations would impact several commercial image classifiers (Mueller et al., 2020; http://obereed.net/unreasonable). The implementation was also informed by another visual classification database browser developed for internal uses by Raytheon as part of their system development. The interfaces are shown in Figure 7, below.



	Source	Base image	Transform	Top answer	Correct	Any response	Score
1	Google	scissor5	Original	SCISSORS	TRUE	TRUE	0.756
2	Clarifai	scissor5	Original	SCISSORS	TRUE	TRUE	1.000
3	Watson	scissor5	Original	PAIR OF SCISSORS	TRUE	TRUE	0.990
4	Amazon	scissor5	Original	SCISSORS	TRUE	TRUE	0.952
5	Inception A	scissor5	Original	SCISSORS	TRUE	TRUE	0.963
6	Inception B	scissor5	Original	SCISSORS	TRUE	TRUE	0.984

	Source	Base image	Transform	Top answer	Correct	Any response	Score
1	Google	scissor5	Edge	PRODUCT	FALSE	TRUE	0.602
2	Clarifai	scissor5	Edge	SCISSORS	TRUE	TRUE	0.984
3	Watson	scissor5	Edge	PAIR OF SCISSORS	TRUE	TRUE	0.988
4	Amazon	scissor5	Edge	ORNAMENT	FALSE	FALSE	0.781
5	Inception A	scissor5	Edge	SCISSORS	TRUE	TRUE	0.578
6	Inception B	scissor5	Edge	SCISSORS	TRUE	TRUE	0.753

Figure 7. Two views of an image classifier database browser used by Mueller et al. (2020) that served as a precursor to the DP.

Prototype I: MNIST Digits using SVM classifier

The first implementation focused on the MNIST data set. MNIST consists of 60,000 labeled handwritten images, and modern image classifiers have achieved very good performance. The MNIST is widely used, and for our purposes convenient because it has a limited number of categories (10). As a demonstration, we trained a support vector machine classifier on 10,000 cases in the training set using svm (support vector machine) function in the e1071 package of the R statistical computing language, with a linear kernel, nu-classification type, using 28x28=785 independent greyscale pixel features each having up to 255 levels of grey. This model and training set was chosen so that it could achieve reasonably high accuracy while still producing significant errors. With training on this limited set, the system achieved an 89% classification accuracy on a held-out cross-validation set. This demonstrated a reasonably high level of accuracy for the classifier, but because the majority of cases were correctly identified, we sampled 10,000 cases (5000 correct and 5000 error) to populate the discovery platform. For a real system, oversampling errors may not be advisable because it will tend to over-focus on errors, and the platform has ways to select and search error cases only in case these need to be examined.

Pre-calculated Information

To deploy the DP, one needs to pre-calculate several values for each case, which are typically easily available. First, each case must have a labelled ground truth. The DP is not intended for exploring and finding errors in an unlabeled data set, but rather to investigate how it performs on known cases. Next, each case must identify what its best-guess response is, and whether the response is correct. Finally, a rank-ordering and confidence level (such as a posterior probability, softmax value, or rank ordering) of labels is used to find similar cases. The DP limits these confidence ratings to a maximum of 10 cases per target image. For the MNIST dataset, these were all of the outcome labels, but for another data set with more labels (such as the fgvc image set discussed in Round 2), each image can define which 10 cases are the most similar. The DP must have access to images in the data set. Initially, those images are hosted via the R Shiny app/webserver. If images are available via another website, the system could be adapted to point to external images as well.

On-demand Calculations and Operations

When the web site session is loaded by a user, the DP reads data tables including the information about each case. We have found that with reasonable hardware, a data set of 10,000 cases can be explored with little timelag. Because the Shiny app maintains a single session for most operations, it will typically only read in the data tables when first run.

On several of explorer tabs, the DP provides tables of examples that match various combinations of target and response. This filtering and sampling is done on-demand, but for our test cases with 10,000 records, is completed almost instantaneously. These tabs also provide means of rank-ordering displayed images based on similarity to the selected ground truth or label. This rank-ordering done with a relatively simple algorithm that calculates and rank-orders via Kullback-Leibler information divergence. Here, for the small category set of 10 target categories, this is easily computed in realtime. However, as the number of label categories increase, this leads to an escalation in the size of the matrices needed to calculate these values, so that with 50 targets, the system could become unstable. We addressed this in the second round of development, and will discuss our strategies for dealing with that below.

Finally, the overview tab includes a correspondence analysis, which relies on eigen decomposition of the confusion matrix. As the size of this matrix increases, it may prove too computationally-intensive to calculate live, although this could be pre-calculated when the system is implemented if that proves burdensome.

Prototype II: Extending for the FGVC Data Set

In collaboration with Raytheon/BBN, we implemented a second discovery platform using the classifier developed by Raytheon/BBN under the XAI program. This involves a substantially larger set of image classes, involving 90 distinct classes of commercial, military, and general aviation aircraft (although many of the classes are different versions of the same base model, such as 737-200 through 737-900). An image classifier was trained that achieved 68.6 % accuracy on these images, and we developed a DP to explore the labelled data set.

Because of the properties of this data set, several interface, data visualization, and algorithm changes were made to the DP, and these in general had no impact or even improved the use of the MNIST version.

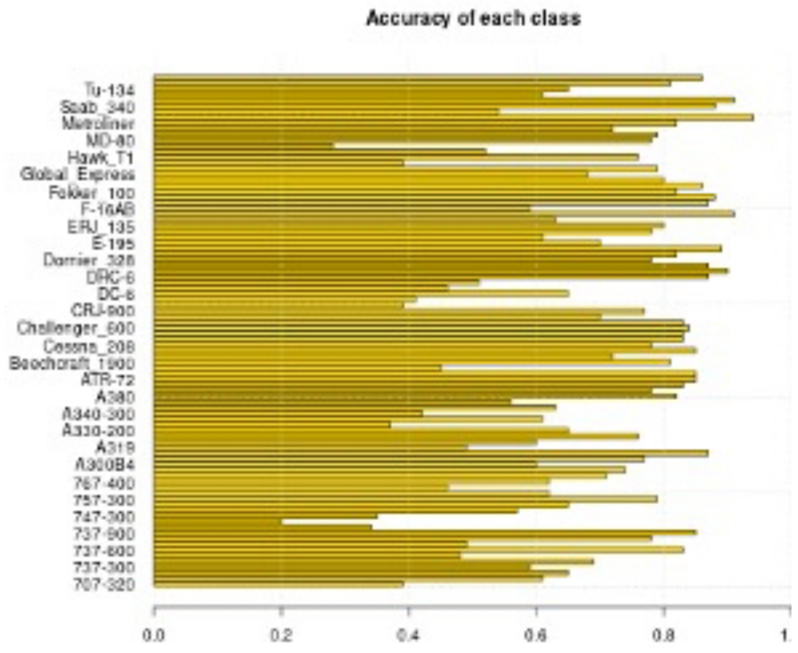
First, the MNIST data set confusion matrix is very dense, because of the small number of categories and the large number of cases. Almost every number class was confused with every other number class at least a few times. In contrast, FGVC is very sparse—most confusions between aircraft are never committed by the AI. This means that many of the visualizations are cluttered with numerous combinations of ground-truth/output label that never occur, making it difficult to even read the labels. As a consequence, we adapted several of the visualizations to filter out empty cells and focus only on the committed errors, which made the histograms easier to read and the axis labels more legible.

Second, because of the larger number of labels, calculating similarity based on error/confidence distributions became unstable. This appears mostly because the web front-end supplied by shiny would timeout before the back end could complete large matrix operations. After some debugging, we traced this to memory issues rather than computation time itself—the matrix representing confidence that is required to calculate error similarity in real time quickly surpassed available memory, which led to poor performance. Because this matrix is actually mostly empty (with only 10 columns being used in any row), we shifted memory storage to a sparse-matrix data structure provided by the sparseM package in R. This appears to have addressed the lag issues we experienced. However, larger data sets may still cause problems, and in those cases, one may need to simply abandon the goal of rank-ordering cases by error similarity. In our estimation, it is not clear whether the ranked exploration provides actionable information (and we have done no user testing to establish it is useful), and these calculations might just be removed from the platform should the calculations prove too burdensome.

Related to the sparsity of the labels, we found that many individual image classes resulted in just a few labels out of the roughly 60 cases. Thus, when trying to select interesting ground-truth/label comparisons, most of the combinations had no matching cases, resulting in a frustrating experience. To address this, we filtered response cases to only those with legitimate examples in the database, and further provided a count of the number of cases in each comparison. This provided an additional advantage insofar is it permits easily finding the most common mistakes made, and helps understand how many cases of each error exist in the data set.

Finally, a number of host-side parameters were added to permit browsing different sizes of images. The MNIST images were 28x28 pixels, which were displayed nicely in a web table expanding their width to 100 pixels. In contrast, the fgvc imagery were much larger, and 100-pixel wide thumbnails did not provide sufficient detail, so we allowed the size of the tables (both in pixels and number of columns) to be specified in the host configuration file.

Figure 8 shows some screenshots of different panels of the DP for the FGVC data set.



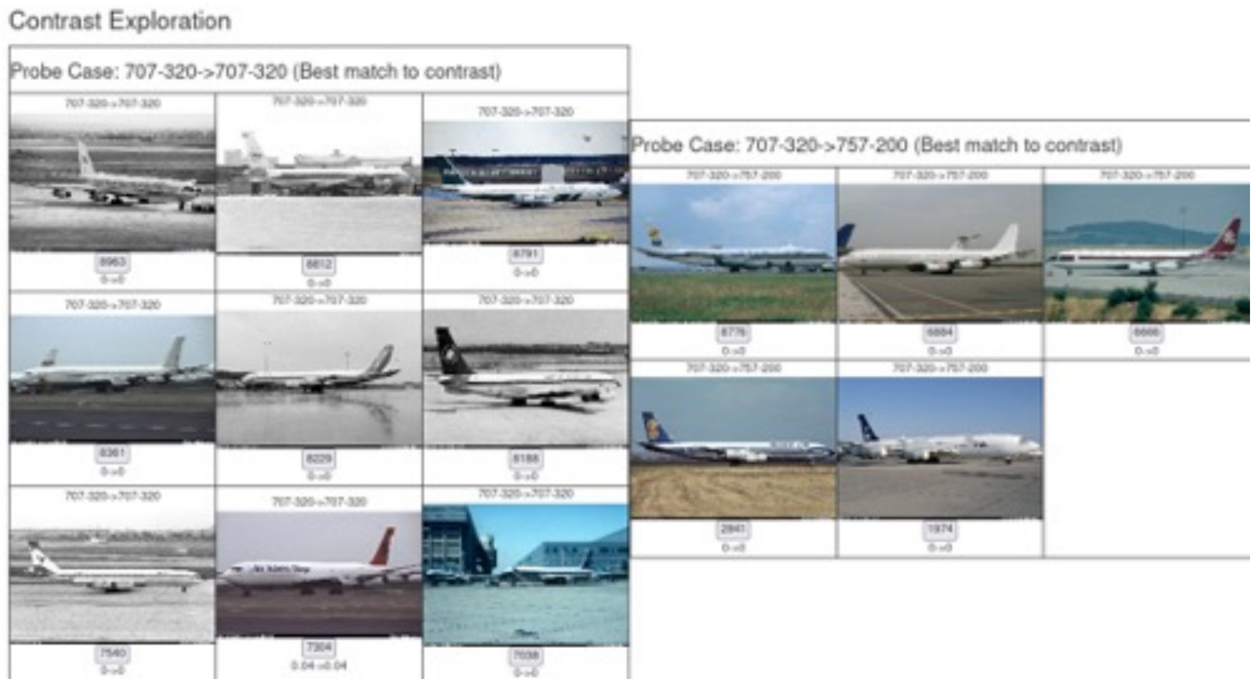

Contrast Exploration
Probe Case: 707-320->707-320 (Best match to contrast)
Probe Case: 707-320->757-200 (Best match to contrast)

Figure 8. Two outputs from the fgvc Discovery Platform. The top panel shows an accuracy plot on the overview pane. The bottom shows a contrast plot comparing correct vs incorrectly identified 707-320 aircraft.

Insights From the Discovery Platform

We have not performed a formal evaluation of the DP. However following the implementation of the fgvc platform, the developer team used the platform to make informal investigations of typically errors and confusions. Patterns that were noticed include:

- The A380 is confused for a hawk_t1 four times. These are very different airplanes (hawk_t1 is essentially an airshow stunt plane, and a 380 is a jumbo jet). Looking at the four confusions, they are all angled in-flight images similar to airshow images often seen of hawks maneuvering.

- hawk_t1 is confused four times for the eurofighter_typhoon. The hawk is usually in red airshow livery. Two of the confusions were when there were many flying in formation (meaning it was unclear what the plane really was), and one was a black and white photo and in cammo livery. The typhoon is a fairly standard fighter jet, and is mostly in grey colors. I noticed also that the hawk is usually pictured on the ground, while the typhoon is often in the air. The last hawk_t1 → typhoon confusion (#4067) is in the air, and sort of washed-out color. All this suggests that incidental features (livery, context, formation flying) are influencing the classification decision.

- E190 and E195 are close in confusion space in the correspondence analysis page, but E190-195 confusions happened only 2 times out of 100, and e195 → e190 24 times out of about 100. These planes are almost identical, the E195 is a couple feet longer but that is the only difference. It is interesting that there is a bias (maybe more of the E190s existed in the training set), and also very interesting that accuracy is still very high --- e190 is 90% and e195 is 70%. So they are very discriminable by the algorithm. It is interesting that the algorithm performs as well as it does, but is probably not due to shape alone, because the angle of the images which determine the perceived length vs height differ a lot for both. Some of the accuacy might come from livery/branding-- maybe 195s were mostly operated by a different set of airlines than the 190s, and that is enough of a signal.

- Within a common airplane class such as the 737, there are a lot of confusions among the subclasses (-200, -300, to -900, etc.). The -500 is most often confused for the -600. The confusions don't seem symmetric, and don't seem to follow a rule like "confused for the larger plane".

- Many prop-based planes get confused occasionally for the DHC-6, a small prop-plane with seven rows of windows. The same is true for DC-6, a larger 4-prop passenger plane. Exploring errors among these show low-level confusions among a lot of propeller-driven aircraft in general. The impression is that jets are more easily distinguishable by the algorithm, but prop planes are more easily confused. Also, there are some large similarity-based errors within prop-planes, like dornier-328→DHC-8-100. These are very similar aircraft, although they can be easily distinguished by their tail cone, but the AI doesn't seem to use that, because the AI makes a lot of errors between these.

These sorts of insights about common errors and mistakes the classifier made and help the user imagine generate reasons for typical errors (color, pose, etc.). They also led to a number of hypotheses bout what the AI was or was not using (shape of tail cone, propeller, etc.). These do not provide the kind of why-justifications typically generated in algorithmic explanations, but may help identify other kinds of patterns that might otherwise be hidden.

Implementation Details

About the DP System

This system provides filtering and display functions that let developers or users browse a set of images labeled with a machine learning classifier. It allows filtering, establishing patterns, and contrasting different conditions to permit discoveries about the AI. In its initial format, it serves as a means for explaining AI through self-explanation and discovery of a user. If heatmap images (e.g., LIME or GRADCAM, etc.) are also available for the image set, these may alternatively be used instead of the original images. With minor changes to the code, both original and heatmap images could be supported.

Implementation

The DP was developed with shiny, an interactive web system based on the R statistical computing language. It must be run using a special server, although RStudio supports running shiny apps via local computers for testing.

Implementing the Discovery Platform

Implementing the Discovery Platform requires a set of images to display, stored within the images/ subdirectory of the www/ directory. These should be in a format that can be displayed directly in html (jpg, gif, or png). Three additional files are used for setting up discovery platform. The basic directory/file structure is:

app.R Main shiny code
main.csv Main data table
probs.csv Probability table
info.json Customize the discovery platform
www/   web files
www/fns.js Custom javascript
www/images images of each case to be browsed.

- main.csv This file contains information about the entire set of images in comma-separated format. Columns must be labeled as in the following example:

```
case, classID,  class, labelID,  label, corr, fname

3,  4,  4,  4,  4,  TRUE, ./images/D0/img00000003.png
4,  1,  1,  1,  1,  TRUE, ./images/D0/img00000004.png
7,  1,  1,  1,  1,  TRUE, ./images/D0/img00000007.png
12,  5,  5,  1,  1,  FALSE, ./images/D0/img00000012.png
19,  6,  6,  6,  6,  TRUE, ./images/D0/img00000019.png
```

Case is a unique but arbitrary numeric label for each image. classID is a numeric identifier (1 and above) for the class of the image. class is a text (or number) naming that class. Similarly, labelID and label describe the best label given to the image by the classifier. corr determines whether it is judged as correct (could be coded 0/1 or TRUE/FALSE), and fname indicates the filename where the image representing the case is stored (within the www/ directory.)

- probs.csv This file contains the 10 top labels given to each case by the classifier, with probabilities or strengths describing the relative strength of each alternative. This file must be exactly 20 columns. The first row must be labels, but the labels can be anything and are not used directly. The file must have the same number of rows as main.csv.
-

```
"X1","X2","X3","X4","X5","X6","X7","X8","X9","X10","X1.1","X2.1","X3.1","X4.1","X5.1","
X6.1","X7.1","X8.1","X9.1","X0"
1,2,3,4,5,6,7,8,9,10,4e-04,0.0239,0.0174,0.9179,0.0097,0.0037,0.0069,0.0013,0.0167,0.0022
1,2,3,4,5,6,7,8,9,10,0.976,0.0093,0.0013,3e-04,0.0014,3e-04,0.0012,0.0096,4e-04,1e-04
1,2,3,4,5,6,7,8,9,10,0.9443,0.0101,0.0181,6e-04,0.005,0.0028,0.0033,0.0112,0.0045,2e-04
1,2,3,4,5,6,7,8,9,10,0.38,0.2249,0.0093,0.0238,0.2596,0.0169,0.0221,0.05,0.0071,0.0065
1,2,3,4,5,6,7,8,9,10,0.0047,0.0153,0.0308,0.0129,0.2841,0.6082,0.0038,0.0201,0.0087,0.0113
1,2,3,4,5,6,7,8,9,10,0.0283,0.2543,0.1173,0.0038,0.2903,0.1773,0.0469,0.0275,0.0399,0.0145
1,2,3,4,5,6,7,8,9,10,0.0733,0.0319,0.026,0.0072,0.829,0.0023,0.0161,0.0065,0.006,0.0017
1,2,3,4,5,6,7,8,9,10,0,0.0019,2e-04,7e-04,5e-04,0.9942,1e-04,6e-04,6e-04,0.0012
1,2,3,4,5,6,7,8,9,10,0.0042,0.0015,0.2508,0.0013,0.675,0.0011,0.0254,0.0236,0.0152,0.0019
```

The first ten columns indicate the top label responses given for a case, in any order. The second ten indicate probabilities associated with each label. The order of labels can be different for each row, but the probabilities and the labels must be in the same order for each row of the data. For cases in which fewer than 10 labels are returned by a classifier, use NA to fill the rest of the first ten label columns. The second ten columns can be 0 or NA if not used. If more than ten labels are returned by the classifier, only the top 10 should be entered in a row.

The labels in this file must start with 1, and correspond to the numbers in labelID column of main.csv.

- info.json This is a standard .json file that contains some of the branding/information to allow customizing the discovery platform without editing the .R code.

```
{
 "title":"XAI Discovery Platform | MNIST Sample Data",

"subtitle":"Michigan Technological University | MacroCognition | IHMC",
 "contact":"Shane T. Mueller (shanem@mtu.edu)",
 "about":"About: MNIST data with linear SVM classifier trained on 10,000 cases. Classifier achieved 89.2% accuracy on training set, and a similar accuracy (89.2%) was achieved for set of 50,000 validation cases. This platform browses 10,000 validation cases, sampled randomly so that 5,000 were correct and 5,000 were errors",
 "otherFeatures":[],
 "classNames":["1","2","3","4","5","6","7","8","9","0"],
 "labelNames":["1","2","3","4","5","6","7","8","9","0"]
}
```

Along with titles and contact, there are three additional optional settings. classNames and labelNames specify the order that the levels of class and label should be displayed in, and must contain all levels found in main.csv of each of these columns. If these are not provided, the default order will be used, which is likely to be alphabetical. Next, otherfeatures indicates any additional feature columns (to be included in main.csv) that can be used for filtering and sorting. This might be a set of higher-level categories or classifications that would be useful to compare on. Support for using these is fairly limited.

Demonstration Site

The code includes a .R file called a build-discovery-platform.R, which will create a simple SVM classifier on a subset of the MNIST handwritten letter digits. I recommend you use RStudio to run this, which is downloaded and installed separately from the main R software.

To build the demonstration site, you must download the MNIST data as a .csv file from Kaggle at https://www.kaggle.com/oddrationale/mnist-in-csv. Only the file mnist_train.csv is used, and it must be unzipped/saved directly in the archive/ subdirectory (the zip file has its files in an archive directory already.) This code requires several libraries to use. It will load this data, create a svm classifier based on 10,000 cases (of the 60,000 total), and then classify both the training and remaining 50,000 cases, on which it achieves an accuracy of about 90%. Training this model and classifying the rest of the cases can take 30 minutes or more depending on your computer. Then, a sample of 10,000 cases (5,000 correct and 5,000 error) are used for the discovery platform, and the properly-formatted main.csv and probs.csv files are generated. Finally, at the end (if this is enabled), each case will be saved as a .png image within www/images, spread across subdirectories with at max 1000 files per directory.

Once this file run is completed successfully, you should be able to run app.R directly in RStudio with the 'Run App' button at the top of the editing pane. This will run the site within RStudio, but you can open in a web browser window by clicking on the 'Show in new window' button in the R Studio Viewer. If you want others to access this server, you must run from a shiny server, which is possible via appropriate software installation on a linux server. Several companies offer shiny app hosting for affordable rates.

Limitations and Potential Future Features

Database Size

Because the 'selected case' pulldown is generated from the data, this pulldown is limited to the first 1000 cases. Creating that pulldown appears to be the main computationally-intensive operation, at least for our test cases with a limited number of labels (10). For systems with 100s or 1000s of categories, some of the bargraph visualizations might break down, and these may need to be adapted or removed to allow the system to work.

Input Format

Currently, the data are read in via a specially-formatted .csv file. R and shiny offer capabilities to read directly from databases, but this is not supported in the current system. R reads in the entire data table once when the page is loaded, and will not read again unless the page is reloaded, so the server-side burden is fairly limited. Extremely large databases could limit system performance, but the MNIST system with 10,000 cases provides a reasonable size to explore, especially when errors are over-sampled.

Other Visualizations

Currently, only a single visualization is supported, and it should be thumbnailable in a 100x100 window. Currently, clicking on the image opens a pop-up window, and this could support multiple views (heatmap overlays), but that capability would have to be added, perhaps with additional columns in main.csv. Something like LIME, GRADCAM, or the like could be incorporated fairly easily in that fashion.

Selecting Similar Examples

Currently, the 'most similar' cases are selected based on Kullback-Leibler distance between cases in the confusion space. Models may permit identifying a similarity space and neighbors directly, but this type of similarity is not currently used. Furthermore, this kind of similarity may require substantial realtime computation to work appropriately.

Incorporating Additional Features

Many times, the input database may have been coded with additional features other than just the ground truth category and the AI classification label. This is only supported via filtering view, but might be incorporated in the categories and contrasts views more directly.

Non-ground-truth Cases

Currently, the system assumes you have a labeled set of images where each ground truth is a single objective category. Sometimes, you may have cases different levels of agreement from multiple human coders or with multiple true labels (40% said woman, 40% said lady, 20% said human). The system currently does not incorporate multiple true labels or display different levels of agreement, but this could be added as additional columns in main.csv and this might be exposed via the UI.

Classification Hierarchies

Many classifiers provide a hierarchy of labels (product|tool|hammer|ball-peen hammer) with different strengths of response. Currently, these can be handled as alternative labels, or by pre-processing with ontologies to exclude alternatives that are hyponyms or hypernyms. Correctness is coded in the data file, so it is possible to score a classification is correct even if its best-matching label is NOT (e.g., calling a hammer a 'product'). However, the UI may not make this obvious.

Video

It may be possible to encode video thumbnails as animated .gif files, although this could be distracting. An alternative is to make video available via another filename specified in a second column of main.csv, and provide javascript to open a control window to view this.

Non-imagery (text, audio, other)

This is likely to not work well, although special-purpose web interfaces could be developed for some of these, provided a thumbnail of the asset can be created and used in the discovery platform.

URL Parameters

Shiny supports processing with web page url parameters, but the current platform avoids. We have experimented with this, and it is possible to incorporate, but tricky to get right. This would permit linking to specific cases by a direct URL.

Acknowlegements

The authors thank Alex Montes de Oca, Jeff Miller, and Joshua Fasching at Raytheon for making the fgvc classifier results available, and Ben Shniederman for suggestions on the discovery platform interface.

References

Mueller, S. T., Agarwal, P., Linja, A., Dave, N., & Alam, L. (2020, December). The Unreasonable Ineptitude of Deep Image Classification Networks. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 64, No. 1, pp. 410-414). Sage CA: Los Angeles, CA: SAGE Publications.