Neo: Generalizing Confusion Matrix Visualization to Hierarchical and Multi-Output Labels

Jochen Görtler*
University of Konstanz
Konstanz, Germany
jochen.goertler@uni-konstanz.de

Fred Hohman
Apple
Seattle, WA, USA
fredhohman@apple.com

Dominik Moritz
Apple
Pittsburgh, PA, USA
domoritz@apple.com

Kanit Wongsuphasawat
Apple
Seattle, WA, USA
kanitw@apple.com

Donghao Ren
Apple
Seattle, WA, USA
donghao@apple.com

Rahul Nair
Apple
Heidelberg, Germany
rahul_nair@apple.com

Marc Kirchner
Apple
Heidelberg, Germany
marc_kirchner@apple.com

Kayur Patel
Apple
Seattle, WA, USA
kayur@apple.com

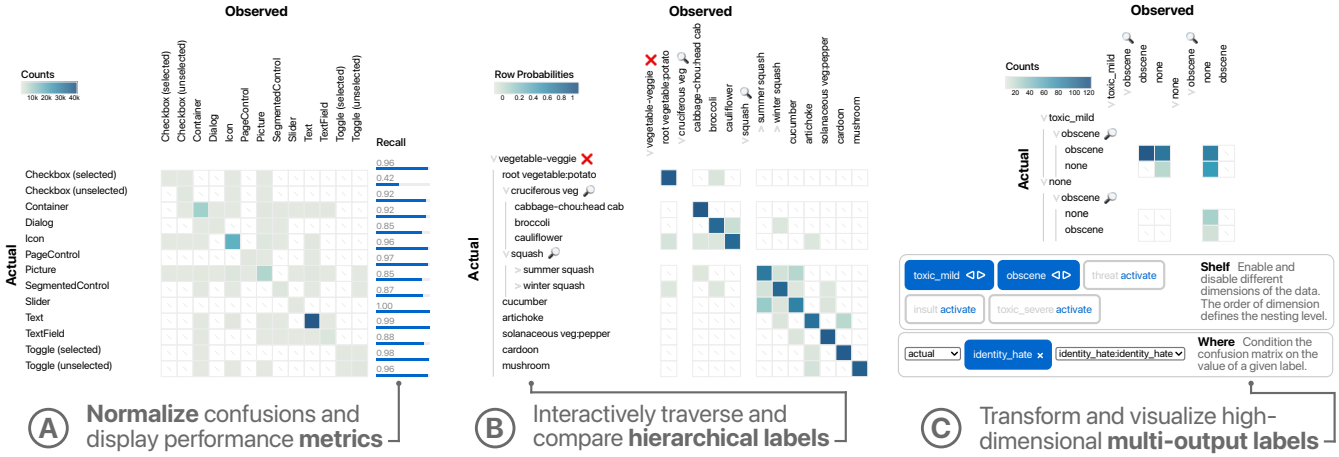

Figure 1: Neo generalizes conventional confusion matrices and enables machine learning practitioners to find hidden confusions, visualize per class metrics, traverse hierarchical labels on tiered axes, and transform high-dimensional, multi-output labels for model evaluation. (A) This confusion matrix for an object detection model computes and shows user-specified performance metrics, such as recall, per class alongside the matrix visualization. (B) This sub-hierarchy of a confusion matrix for a 1,000-class image classifier compares the confusions of different vegetables, such as *cucumber* and *mushroom*, against the sub-hierarchies of *cruciferous vegetables* and *squash*. (C) This confusion matrix for a naive multi-output online toxicity detector conditions and filters by *identity hate* confusions, nests *obscene* confusions under *mild toxic* confusions, and finds that the model misses to identify many *obscene* comments.

ABSTRACT

The confusion matrix, a ubiquitous visualization for helping people evaluate machine learning models, is a tabular layout that compares predicted class labels against actual class labels over all data instances. We conduct formative research with machine learning practitioners at Apple and find that conventional confusion matrices do not support more complex data-structures found in modern-day applications, such as hierarchical and multi-output labels. To express such variations of confusion matrices, we design an algebra that models confusion matrices as probability distributions. Based on this algebra, we develop Neo, a visual analytics system that

*Work done at Apple.

This work is licensed under a Creative Commons Attribution International 4.0 License.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9157-3/22/04.
https://doi.org/10.1145/3491102.3501823

Konstanzer Online-Publikations-System (KOPS)
URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-2-1xpfvhkvg89m35

enables practitioners to flexibly author and interact with hierarchical and multi-output confusion matrices, visualize derived metrics, renormalize confusions, and share matrix specifications. Finally, we demonstrate Neo's utility with three model evaluation scenarios that help people better understand model performance and reveal hidden confusions.

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; **Visual analytics**; • **Computing methodologies** → *Machine learning*; *Artificial intelligence*.

KEYWORDS

Confusion matrices, model evaluation, visual analytics, machine learning, interactive systems

ACM Reference Format:
Jochen Görtler, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, Donghao Ren, Rahul Nair, Marc Kirchner, and Kayur Patel. 2022. Neo: Generalizing Confusion Matrix Visualization to Hierarchical and Multi-Output Labels. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3491102.3501823

1 INTRODUCTION

Machine learning is a complex, iterative design and development practice [4, 24], where the goal is to generate a learned model that generalizes to unseen data inputs. One critical step is *model evaluation*, testing and inspecting a model's performance on held-out test sets of data with known labels. Due to the size of modern-day machine learning applications, interactive data visualization has been shown to be an invaluable tool to help people understand model performance [5, 13, 25, 36].

A ubiquitous visualization used for model evaluation, particularly for classification models, is the *confusion matrix*: a tabular layout that compares a predicted class label against the actual class label for each class over all data instances. In a typical configuration, rows of the confusion matrix represent actual class labels and the columns represent predicted class labels (synonymously, these can be flipped via a matrix transpose). These visualizations are introduced in many machine learning courses and are simultaneously used in practice to show what pairs of classes a model confuses. Succinctly, confusion matrices are the "go-to" visualization for classification model evaluation.

Despite their ubiquity, conventional confusion matrices suffer from multiple usability concerns. Confusion matrices show a visual proxy for accuracy (e.g., entries on the diagonal of the matrix), which alone has been shown to be insufficient for many evaluations [39]. Furthermore, the diagonal of a confusion matrix often contains many more instances than off-diagonal entries (can be orders of magnitude), which hides important confusions (i.e., off-diagonal entries). As practitioners improve their model, the net effect moves off-diagonal instances to the diagonal, further exacerbating this problem of hiding confusions. Ironically, the better the model optimization, the harder it is to find confusions. Confusion matrices also suffer from scalability concerns, e.g., when a dataset has many classes, has strong class imbalance, has hierarchical structure, or has multiple outputs.

We believe confusion matrices can be significantly improved to help practitioners better evaluate their models. To understand specific challenges around using confusion matrices, we conducted a formative research study through a survey with machine learning practitioners at Apple. We found that in many machine learning applications confusion matrices become cumbersome to use at scale, do not show other metrics model practitioners need to know (e.g., precision, recall), and can be hard to share. Moreover, confusion matrices only support flat, single-label data structures; more complex yet common hierarchical labels and multi-output labels are not supported.

Informed by findings from the formative research and a literature review, we create a *confusion matrix algebra*, which models confusion matrices as probability distributions and produces a unified solution for pitfalls of conventional confusion matrices. Based on this algebra, we design and develop Neo, a visual analytics system that enables practitioners to flexibly author and interact with confusion matrices in diverse configurations with more complex label structures. The design of Neo extends the confusion matrix, allowing users to visualize additional metrics for analysis context, inspect model confusions interactively through multiple normalization schemes, visualize hierarchical and multi-output labels, and easily share confusion matrix configurations with others. Neo maintains the familiar format of confusion matrices, using a conventional confusion matrix as the basis of the visualization.

In this work, our contributions include:

- **Formative research**, including common challenges and analysis tasks, from surveying machine learning practitioners at Apple about how confusion matrices and model evaluation visualizations are used in practice.
- **A confusion matrix algebra** that generalizes and models confusion matrices as probability distributions.
- **Neo**,[1] **a visual analytics system** for authoring and interacting with confusion matrices that supports hierarchical and multi-output labels. Neo also introduces a specification (or colloquially, a "spec") that enables sharing specific visualizations with others. Neo is reactive in that authoring a spec updates the visualization, and interacting with the visualization updates the spec.
- **Three model evaluation scenarios** demonstrating how Neo helps practitioners evaluate machine learning models across domains and modeling tasks, including object detection, large-scale image classification, and multi-output online toxicity detection.

We believe machine learning should benefit everyone. Understanding where models fail helps us correct them to enable better experiences. We hope the lessons learned from this work inform the future of model evaluation while inspiring deeper engagement from the burgeoning intersection of human-computer interaction and artificial intelligence.

[1] https://github.com/apple/ml-hierarchical-confusion-matrix

2 RELATED WORK

Model evaluation is a key step to successfully applying machine learning. However, what it means for a model to perform well greatly depends on the task. A variety of metrics have been developed to evaluate classifiers [16]; common example metrics include *accuracy*, *precision*, and *recall*. However, there is no one-size-fits-all metric,[2] and the utility of metrics depend on the modeling task.

Model Performance Visualizations. The visualization community has developed novel visual encodings to help practitioners better understand their model's performance. These techniques can be categorized as either *class-based* or *instance-based*. For class-based techniques, Alsallakh et al. [1] use a radial graph layout where the links represent confusion between classes. Seifert and Lex [27] embed all test and training samples into a radial coordinate systems where units are classes. Regarding instance-based visualization, Amershi et al. [5] propose a unit visualization that shows how each instance is classified and shows the closeness of instances in the feature space. *Squares* [25] extends this visualization and shows, per class, how instances are classified within a multi-classifier. Similarly, *ActiVis* [18] uses instance-based prediction results to explore neuron activations in deep neural networks. While we focus on practitioners, previous work has also studied confusion matrix literacy and designed alternative representations for non-expert and public algorithmic performance understanding [28]. Common to these works is that they introduce new visualization concepts that may not be familiar to machine learning practitioners and therefore require training and adaption time.

Confusion Matrix Visualizations. Instead of introducing alternative visual encodings, our approach aims to enhance confusion matrices directly, a ubiquitous visualization that already has familiarity within the machine learning community [37], and adapt them to types of data that are encountered in practice today. There exists some work that enhances conventional confusion matrices. For example, individual instances have been shown directly in the cells of a confusion matrix [7, 36]. Alsallakh et al. [2] investigate hierarchical structure in neural networks using confusion matrices. In their work, hierarchies can be constructed interactively based on blocks in the confusion matrix, which are then shown using icicle plots. They also provide group level statistics for the elements of the hierarchy. However, in contrast to our work, their system does not consider multi-output labels.

Confusion matrices are also used in iterative model improvement. Hinterreiter et al. [13] propose a system to track confusions and model performance over time by juxtaposing confusion matrices of different modeling runs. Their system also provides an interactive shelf to specify the individual runs. Our work also features an interactive shelf; however, its purpose in our work is to drill down into sub-hierarchies of a larger confusion matrix. Furthermore, confusion matrices have been used to directly interact with machine learning models. As such, they can be used to interactively adapt decision tree classifiers [35], by augmenting them with information about the splits that are performed by each node. For models that are based on boosting, Talbot et al. [33] propose a system to adjust the weights of weak classifiers in an ensemble to achieve better performance. Furthermore, Kapoor et al. [20] propose a technique to interactively steer the optimization of a machine learning classifier, based on directly interacting with confusion matrices. None of these systems consider hierarchical or multi-output labels.

There are also different approaches that generalize beyond conventional confusion matrices. Class-based similarities from prediction scores instead of regular confusions have been proposed to generalize better to hierarchical and multi-output labels [3]. Zhang et al. [40] embed pairwise class prediction scores into a Cartesian coordinate system to compare the performance of different models. Furthermore, multi-dimensional scaling has been used to embed confusion matrices into 2D [32]. In contrast to our work, these adaptions stray further from familiar confusion matrix representations.

Model Confusions as Probability Distributions. Framing the confusion matrix as a probability distribution has been used by the machine learning community to investigate classifier variability [8]. In addition, other work shows how a probabilistic view of the confusion matrix can be used to quantify classifier uncertainty [34]. However, both these works only consider binary classification. Preliminary work on generalizing to multi-label problems [22] computes the contribution of an instance to a cell, but only if the prediction was only partially correct. Our work builds upon these views and produces a unified language for generalizing confusion matrices to hierarchical and multi-output labels.

Table Algebra. Our work is inspired by *relational algebra theory* [10] and the *table algebra* in *Polaris* [29], now the popular software *Tableau*, and its work on visualizing hierarchically structured data [30]. In Polaris, a user can visually explore the contents of a database by dragging variables of interest onto "shelves". The contents of a shelf are then transformed into queries to a relational database or OLAP cube, which retrieves the data for visualization. Our approach is different in that we support operations on matrices and that it is based on probability distributions rather than a relational database model.

3 FORMATIVE RESEARCH: SURVEY, CHALLENGES, & TASKS

To understand how practitioners use confusion matrices in their own work, we conducted a survey that resulted in 20 responses from machine learning researchers, engineers, and software developers focusing on classification tasks at Apple. Respondents were recruited using an internal mailing list about machine learning tooling and targeted practitioners who regularly use confusion matrices. We take inspiration from the methods used in previous visualization literature on multi-class model visualization [25], bootstrapping our survey questions from their work. The survey consists of eight questions centered around machine learning model evaluation and confusion matrix utility. The first two questions (**Q1** and **Q2**) are multiple choice, while the remaining (**Q3–Q6**) are open responses.

Q1. Which stages of machine learning do you typically work on?
Q2. How many classes does your data typically have?
Q3. When do you use confusion matrices in your ML workflow?
Q4. Which insights do you gain from using confusion matrices?

[2]There is no better illustration of this than viewing the overwhelming number of different metrics that can computed from a confusion matrix: https://en.wikipedia.org/wiki/Confusion_matrix#Table_of_confusion.

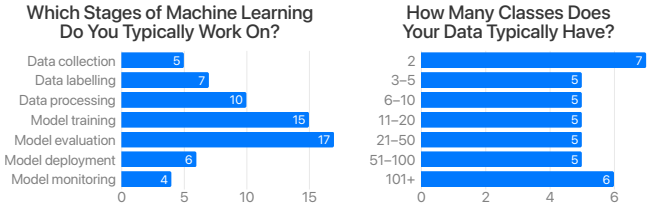

Figure 2: Survey responses from machine learning practitioners (multiple choice questions). Left: respondents cover every stage of machine learning process; many of them work on "data processing" and "model training," with the majority of respondents indicating "model evaluation," the specific machine learning stage we focus on in this work. Right: respondents work on classification models of a variety of sizes, ranging from binary classifiers to models with over 1,000 classes.

Q5. Which insights are missing, or you wish you would also gain from using a confusion matrix?
Q6. How often are your labels structured hierarchically (for example *apple* could be in the category *fruit*, which then is part of the category *food*)? How do you work with hierarchical confusions? How deep are the hierarchies?
Q7. When do you encounter data where one instance has multiple labels (for example an instance that is *apple* and *ripe*)? How many labels are typically associated with an instance?
Q8. How else do you visualize your data and errors besides confusion matrices? What are their advantages?

From the survey data, we used thematic analysis to group common purposes, practices, and challenges of model evaluation into categories [11]. Throughout the discussion, we use representative quotes from the respondents to illustrate the main findings.

3.1 Respondents' Machine Learning Backgrounds

We asked practitioners what stages of the machine learning process they typically work on (**Q1**, multiple choice), to establish context about the respondents' backgrounds. The left-hand side of Figure 2 shows a histogram of what stages our respondent's have experience in, sorted by the chronological order of stages in the machine learning development process [4]. With some expertise represented at every stage of machine learning development, most respondents indicate their work falls between "data processing," "model training," and "model evaluation." This diverse experience and concentration on model evaluation give us, the researchers, confidence that the population of practitioners surveyed contains the relevant experience and knowledge to speak about the intricacies of model evaluation with confusion matrices.

To gain insight into the scale of the respondents' modeling work, we also asked about the typical number of classes modeled from their datasets (**Q2**, multiple choice). The right-hand side of Figure 2 shows a histogram for these responses, sorted from the fewest number of classes (binary classification) to the largest (101+). Results show an emphasis on binary classification, a majority skew towards models with fewer than 50 classes, but also representation from larger-scale models with over 100 classes. These results establish that our respondents have worked with small-scale datasets, large-scale datasets, and everything in-between, strengthening our confidence that many different machine learning applications are represented in our formative research.

3.2 Why Use Confusion Matrices?

We first categorize and describe the reasons why respondents use confusion matrices (**Q3**). While we expected certain use cases to be reported, we were surprised by the number of roles and responsibilities confusion matrices satisfy in practice, such as performance analysis, model and data debugging, reporting and sharing, and data annotation efforts (**Q4**).

3.2.1 Model Evaluation and Performance Analysis. Confusion matrices are constructed to evaluate, test, and inspect class performance in models; therefore, it is unsurprising that most of the responses, 14/20, indicate that model evaluation is the main motivation for using confusion matrices in their own work. Respondents explain that detailed model evaluation is critical to ensure machine learning systems and products produce high-accuracy predictions or a good user experience. According to one respondent, confusion matrices allow a practitioner to see *"performance at a glance."* One frequent and primary example reported was checking the presence of a strong diagonal; diagonal cells indicate correctly predicted data instances (whereas cells outside the diagonal represent confusions), therefore a strong diagonal is found in well-performing models.

3.2.2 Debugging Model Behavior by Finding Error Patterns. Besides seeing performance at a glance, 7/20 respondents indicated that confusion matrices are also useful for identifying error patterns to help debug a model. Regarding pattern identification, a respondent said confusion matrices *"allow me to see how a certain class is being misclassified, or if there is a pattern in misclassifications that can reveal something about the behavior of my model."* Respondents described multiple common patterns practitioners look for, including checking the aforementioned strong diagonal of the matrix, finding classes with the most confusions, and finding classes that are over-predicted. Another interesting pattern reported by a natural-language processing practitioner was determining the directionality of confusions for a pair of classes. For example, in the case of a bidirectional language translation model, does a particular sentence correctly translate from the source to the target language, but not the reverse. These patterns can be *"...much more revealing than a simple number,"* and help practitioners find shared similarity between two confused classes.

3.2.3 Communication, Reporting, and Sharing Performance. While confusion matrices help an individual practitioner understand their own model's behavior, they are also used in larger machine learning projects with many invested stakeholders. Here, it is critical that team members are aware of the latest performance of a model during development, or monitoring the status of a previously-deployed model that is evaluated on new data. One respondent reported that *"exporting the matrix is more useful,"* since confusion matrices are commonly shared in communication reports with other individuals.

3.2.4 High-quality Data Annotation. Beyond model evaluation, respondents said confusion matrices are also useful for data labelling/annotation work. Machine learning models require large datasets to better generalize, which results in substantial efforts to obtain high-quality labels. In this use case, a practitioner wants to understand annotation performance instead of model performance; in some scenarios the same practitioner fulfills both roles.

Some newly labelled datasets undergo quality assurance, where a subset of the newly labelled data is scrutinized, adjusted, and corrected if any labels were incorrectly applied. These labels are then compared against the original labelled dataset using a confusion matrix. These data label confusion matrices visualize the performance of a data annotator (could be human or computational) instead of a model's performance (which can also be thought of as an annotator). This process allows practitioners to find data label discrepancies between different teams. For example, one respondent reported they *"get to understand if there are certain labels or prompts that are causing confusion between the production [label] team and the quality assurance [label] team."* The quality assurance team often shares these visualizations with the production labeling teams to *"improve the next [labeling] iteration,"* guiding annotation efforts through rich and iterative feedback.

3.3 Challenges with Confusion Matrices

When prompted about where confusion matrices may not be sufficient (**Q5**), respondents voiced that they have experienced challenges (**C1–C4**) due to limitations with its representation and lack of support for handling more complex dataset structure. Visualizations for these datasets either did not exist or were shoehorned into existing confusion matrices by neglecting or abusing label names and structure (**Q8**).

*3.3.1 Hidden Performance Metrics (**C1**).* The most common limitation of conventional confusion matrices discovered from our survey is their inability to show performance metrics for analysis context. Over half, 11/20, respondents said that it was important to see other metrics alongside confusions (**Q8**). Even accuracy is not explicitly listed in a confusion matrix but must be computed from specific cells for each class, which can be taxing when performing the mental math over and over. While respondents listed other important performance metrics such as precision, recall, and true/false positive/negative rates, deciding which metrics are important is specific to the modeling task and domain. Lastly, when sharing confusion matrices with others, respondents said it is important to provide textual descriptions of performance to help focus attention on specific errors.

*3.3.2 Complex Dataset Structure: Hierarchical Labels (**C2**).* Another big challenge for confusion matrices is capturing and visualizing complex data structures that are now common in machine learning applications. Conventional confusion matrices assume a flat, one-dimensional structure, but many datasets today across data types have hierarchical structure. When asked specifically about dataset structure, 9/20 respondents said they work with hierarchical data and that typical model evaluation tools, like confusion matrices, do not suffice (**Q6**). For example, an *apple* class could be considered a subset of *fruit* which is a subset of *food*. One respondent indicated that their team works almost exclusively with hierarchical data. In the applications with hierarchical data, respondents indicated that the hierarchies were on average 2–4 levels deep (i.e., from the root node to the leaf nodes). Handling hierarchical classification data and the subgroups inherent to its structure is currently not supported in confusion matrix representations

*3.3.3 Complex Dataset Structure: Multi-output Labels (**C3**).* Another type of dataset structure complexity is also well-represented in our survey, namely datasets that have instances with multi-output labels (**Q7**). For example, an *apple* could be *red* and *ripe*. Over half, 11/20, of the respondents indicated that they work with datasets with multi-output labels that conventional confusion matrices do not support. In such datasets, respondents said that, on average, data instances have 1–3 labels each, but one respondent described an application where instances had 20+ labels. It is important to also note the distinction between labels and metadata: metadata is any auxiliary data describing an instance, whereas a label denotes a specific model output for prediction. In short, all labels are metadata, but not all metadata are labels.

*3.3.4 Communicating Confusions while Collaborating (**C4**).* We have already identified and discussed the need for communicating model performance and common confusions in collaborative machine learning projects. However, there remains friction when sharing new model results with confusion matrices, for example, a loss of quality and project context (e.g., copying and pasting charts as images into a report). It can be time consuming for a practitioner to prepare and polish a visualization to include in a report, yet it is important to ensure model evaluation is accessible to others. Some respondents said it would be convenient if their confusion matrices could be easily exported. This challenge is twofold: what are better and sensible defaults for confusion matrix visualization, and how can systems reduce the friction for practitioners sharing their latest model evaluations?

3.4 Motivation and Task Analysis

From our formative research, there is clear opportunity to improve confusion matrix visualization. Practitioners reported that conventional confusion matrices, while useful, are insufficient for many of the recent advancements and applications of machine learning, and expressed enthusiasm for visualization to better help understand model confusions. This research also yielded several key ideas that inspired us to rethink authoring and interacting with confusion matrices. To inform our design, we distill tasks that practitioners perform to understand model confusions. Tasks (**T1–T4**) map one-to-one to challenges (**C1–C4**):

- **T1**. Visualize derived performance metrics while enabling flexible data analysis, such as scaling and normalization (**C1**).
- **T2**. Traverse and visualize hierarchical labels (**C2**).
- **T3**. Transform and visualize multi-output labels (**C3**).
- **T4**. Share confusion matrix analysis and configurations (**C4**).

4 CONFUSION MATRIX ALGEBRA

From our formative research, we aim to generalize confusion matrices to include hierarchical and multi-output labels. For these

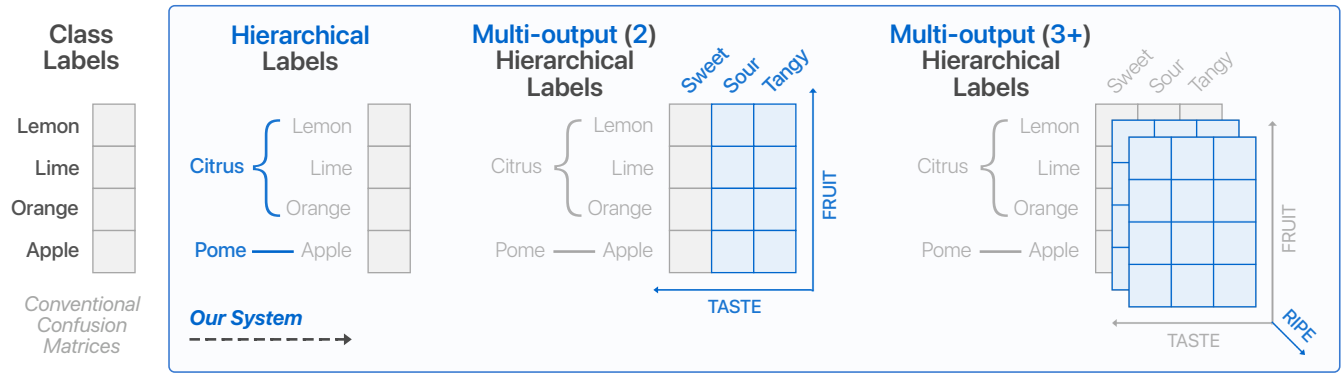

Figure 3: A visual representation of class labels for conventional confusion matrices (left) compared to our work (in blue) that supports hierarchical labels and multi-output labels. To build a confusion matrix from any of these label structures, compute every combination of the actual label against the predicted label for all classes.

types of data, analysis usually requires data wrangling as a preprocessing step, where practitioners develop one-off scripts. Our work takes a different approach: We provide a unified view of the different operations and analysis tasks for confusion matrices in the form of a specification language (**T4**) that is based on a key insight: ***Confusion matrices can be understood as probability distributions***. While this way of viewing confusion matrices may seem unwieldy at first, its expressiveness becomes clear when we think about how practitioners interact with hierarchical and multi-output labels (Figure 3).

Confusion matrices show the number of occurrences for each combination of an actual class versus a predicted class. Rows in a confusion matrix represent actual classes, columns represent predicted classes, and the cells represent the frequencies of all combinations of actual and predicted classes. Our algebra leverages that the actual class X and the predicted class Y can be viewed as variables in a multivariate probability distribution $P(X, Y)$. The probability mass function of this distribution is given by the relative frequencies of occurrences, which we obtain by dividing the absolute frequencies by the number of instances in the dataset. For an introduction to multivariate probabilities, we recommend the book by Hogg and Tanis [14]. Here we will explain the concepts of our algebra using the labels $Fruit = \{\text{apple}, \text{orange}, \text{lemon}\}$ as an example. In this setting, the following describes a cell in the confusion matrix, specifically *apples* that are mistaken for *oranges*:

$$P(\text{Fruit}_X = \text{apple}\,,\ \text{Fruit}_Y = \text{orange}).$$

This probabilistic framing allows us to use the standard operations of multivariate probability distributions to transform our data. In particular, we use the following operations, which we also illustrate in Figure 4: **Conditioning** primes a probability distribution on given values. We can use this operation to extract sub-views of a larger confusion matrix. **Marginalization** allows us to discard variables of multivariate distributions that we are currently not interested in by summing over all such variables. These operations have the algebraic property that their results are again probability distributions—mathematically this is defined as *closedness*. This property is not purely theoretical, but rather it also has practical implications: It allows us to chain multiple operations together to form complex queries. Moreover, the algebra automatically ensures correct normalization after every step. In addition to the two operations above, we also propose a **nesting** operation, which is useful to investigate multiple labels simultaneously.

4.1 Normalization

Normalization is essential for confusion matrices as it determines how the data is visualized (**T1**). Our probabilistic framework guarantees normalization implicitly, as all objects are probability distributions. Depending on the task, it might make sense to normalize a confusion matrix by rows or columns. Choosing a normalization scheme can emphasize patterns that large matrix entries might otherwise hide (example shown in Section 6.1). Normalizing by rows or by columns also produces *recall* and *precision*, two widely used performance metrics echoed from our formative research. The recall for a label is the value on the diagonal of a matrix normalized by rows: $P(\text{Fruit}_Y = \text{orange} \mid \text{Fruit}_X = \text{orange})$. Similarly, the precision for a label is the value on the diagonal of a matrix normalized by columns: $P(\text{Fruit}_X = \text{orange} \mid \text{Fruit}_Y = \text{orange})$. Both cases can be computed using Bayes' rule.

4.2 Hierarchical Labels

With our algebra, practitioners can understand how confusions relate to hierarchical labels by drilling down into specific sub-hierarchies (**T2**). In addition, we can use the hierarchical structure to improve the visual representation of large confusion matrices by collapsing sub-hierarchies. Collapsing sub-hierarchies is equivalent to summarizing multiple entries. First, we collect all the rows/columns that belong to the category to be collapsed. In terms of probability distributions, we create a compound probability (here for *Citrus*) for these items:

$$P(\text{Fruit}_X = \text{Citrus}\,,\ \text{Fruit}_Y = \text{Citrus}) =$$
$$P(\text{Fruit}_X \in \{\text{lemon}, \text{orange}\}\,,\ \text{Fruit}_Y \in \{\text{lemon}, \text{orange}\})$$

This rewrite is possible because, for visualization, the individual rows/columns of a confusion matrix are not affected by another—they are *mutually independent*. Therefore, we can conclude $P(\text{Citrus}) = P(\text{lemon}) + P(\text{orange})$.

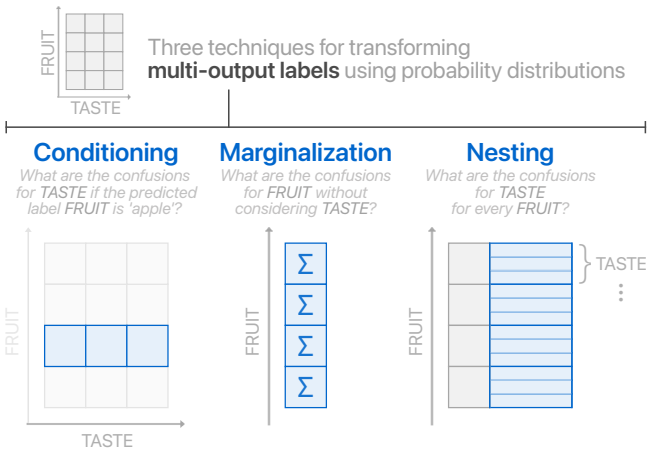

Figure 4: A visual representation of the three techniques for transforming high-dimensional multi-output labels. First, we can condition the confusion matrix based on the value of another label. To focus on a single label, we can use marginalization to sum across ignored labels. We can also nest multiple labels to form hierarchical labels.

The other type of analysis that our algebra supports is drilling down into a sub-hierarchy. For this, we will condition the multivariate distribution on the rows/columns that we want to consider:

$$P(\text{Fruit}_X\,,\ \text{Fruit}_Y \mid \text{Fruit}_X = \text{Citrus}\,,\ \text{Fruit}_Y = \text{Citrus})$$

This operation results in a new confusion matrix that only contains the specified rows and columns as shown in Figure 4.

4.3 Multi-output Labels

Multi-output labels make it significantly harder to evaluate a model's performance. The number of cells in a confusion matrix grows exponentially for datasets with multi-output labels. Adding an additional label *Taste* = {sweet, sour, tangy} to the fruit dataset results in 81 possible combinations of actual and predicted states:

$$|\text{Fruit}_X| \times |\text{Fruit}_Y| \times |\text{Taste}_X| \times |\text{Taste}_Y| = 3 \times 3 \times 3 \times 3 = 81$$

Our algebra provides multiple techniques to transform high-dimensional confusions into 2D for different analyses (**T3**), illustrated in Figure 4. In the following discussion, we use example analysis questions to ground the explanation of each technique.

Initially, an analyst might ask *What are the confusions for "Taste", if the predicted label was "apple"?*, i.e., we consider confusions for one label given a class of a different label. We achieve this in our algebra by conditioning the multivariate distribution on this class. The following example shows the confusion matrix only for *apples*:

$$P(\text{Taste}_X\,,\ \text{Taste}_Y \mid \text{Fruit}_X = \text{apple}\,,\ \text{Fruit}_Y = \text{apple})$$

This operation usually changes the number of columns and rows of the resulting confusion matrix because not all labels necessarily occur together with the fixed label (Figure 4, left).

Furthermore, an analyst may currently not be interested in one of the variables and ask: *What are the confusions for "Fruits" without considering their "Taste"?* In this case, we can discard the needless variable in our probabilistic framework using marginalization. Here, we discard *Taste*:

$$P_{\text{Fruit}}(\text{Fruit}_X\,,\ \text{Fruit}_Y) = \sum_i \sum_j P(\text{Fruit}_X\,,\ \text{Fruit}_Y\,,\ \text{Taste}_X(i)\,,\ \text{Taste}_Y(j))$$

Note that this operation does not change the dimensionality of the variables that we are interested in but instead sums over the frequencies of the discarded entries accordingly (Figure 4, middle).

Finally, analysts that need to understand the relationship between two different variables may ask: *What are the confusions for the "Taste" for every "Fruit"?* To inspect multiple dimensions simultaneously, our algebra can nest one label below another. Multiple labels in a dataset form a high-dimensional confusion matrix, which cannot be readily visualized using a 2D matrix representation. The nesting operation solves this problem by realizing all possible combinations of labels in a structured manner (the *power set* of the variables) and induces a hierarchical structure—the relationship between parent and child is given by the ordering of the nesting (Figure 4, right). This is a useful technique for visualizing joint distributions.

5 NEO: INTERACTIVE CONFUSION MATRIX VISUALIZATION

To put our confusion matrix algebra into practice, we design and develop Neo, a visual analytics system that enables practitioners to flexibly author and interact with confusion matrices for model evaluation. Our visualization system is agnostic to the model architecture and data. If the classification problem (or data annotation task) can record instance labels and predictions, Neo can ingest the results. Throughout the following section, we link relevant views and features to the tasks (**T1–T4**) identified from our formative research (Section 3.4).

5.1 Design Goal: Preserve Familiar Confusion Matrix Representation

Whereas many machine learning visualizations do not have an established form, confusion matrices have an expected and borderline "standardized" representation. Instead of reinventing the confusion matrix visualization, our primary design goal for Neo was to leverage the familiarity of confusion matrices and improve upon their functionality with complementary views and interaction. For example, in the simplest case where a practitioner has a classification model with a dataset whose instances have no hierarchy and only one class label, Neo shows a conventional confusion matrix. However, even in these cases there is still opportunity for improving model evaluation through interaction.

5.2 Specification for Matrix Configuration

Neo is built upon a powerful domain-specific language (DSL) for specifying a confusion matrix configuration. Implemented in Neo using JSON, this paradigm provides similar benefits to other declarative specification visualizations [26]: automated analysis, reproducibility, and portability. Figure 5 shows an example "spec" and its different fields. In this section we describe every field of the spec.

```
{
  "normalization": "total",          I Normalize matrix data
  "encoding": "color",               I Choose visualization encoding
  "collapsed": ["fruit:apple"],      I Hide matrix sub-hierarchy
  "measures":                        |
    "accuracy",                      |
    "precision",                     | Visualize evaluation metrics
    "recall"                         |
  ],                                 |
  "classes": ["fruit"],              I Activate multi-output labels
  "filter": ["fruit:tropical"],      I Zoom into sub-hierarchy
  "where": {                         |
    "qualifier": "actual",           |
    "label": "citrus",               | Condition confusion matrix
    "is": "citrus:orange"            | on value of a given label
  }                                  |
}
```

Figure 5: Neo's JSON specification based on our confusion matrix algebra. The specification configures the confusion matrix based on the selected `normalization` scheme, visualization `encoding`, and desired `measures`, but also saves the shown state of the hierarchy (`collapsed`, `filter`) and multi-output labels using either marginalization (`classes`), nesting (order of `classes` array), or conditioning (`where`).

Neo is a *reactive system*: configuring the spec updates the visualization, and interacting with the visualization updates the spec. This is a powerful interaction paradigm where a practitioner can tailor their desired view using either code or the interface while remaining in sync [21]. Once a practitioner is satisfied with their visualization, they can easily share their spec with others since their view is represented as a JSON string (**T4**). In Neo, the spec is hidden by default, but is exposed through a single button click.

5.3 Interacting with Confusion Matrices

The primary view of Neo is the confusion matrix itself (multiple examples seen in Figure 1). Rows represent actual classes and columns represent predicted classes. A cell contains the number of data instances incorrectly predicted from the row class as the column class; the exception are cells along the diagonal that indicate the number of correctly predicted instances for a particular class. To see how many instances are in each cell, one can hover over any cell to display a textual description of the confusion count. This feature was requested in our formative research. Moreover, hovering over a cell highlights its row and column in the matrix, using a light amber background color (Figure 6A), to ease a user's eye-tracking when reading the axis labels. We chose to use a visual encoding for confusion counts instead of numeric labels within each cell, since with bigger data (e.g., cells with confusions larger than 3 digits) the labels become long, grow the size of each cell, and ultimately inhibit the number of classes can fit in one display. Furthermore, for models with many classes, from our formative research practitioners wanted to a high-level overview of the performance of a model first with the option to inspect specific cells, hence the design of the details-on-demand textual descriptions.

5.3.1 Visualization Encodings and Confusion Normalization. The default encoding is `color` (arguably the default in practice). Users can toggle between a `color` encoding and a `size` encoding where

Figure 6: (A) Brushing a cell in Neo displays the confusion information in a natural language caption. (B) Confusion matrix cells with a 0 value are excluded from the encoding scale. (C) Users can choose between a color and size encoding for the confusion matrix cells.

inner squares are scaled to support comparison of absolute values (Figure 6C); this is set in the specification in the `encoding` field (Figure 5). Regardless of encoding, this common representation already presents a problem with confusion matrices: the diagonal contains many more instances than off-diagonal entries (e.g., orders of magnitude), which hide important confusions in the matrix. As practitioners improve a model over time, the net outcome moves instances from off-diagonal entries to the diagonal, further exacerbating this problem. Ironically, the better the model optimization, the harder it is to see confusions.

Neo addresses this issue in multiple ways. First, Neo leverages a color discontinuity for the value 0 [19]. Cells with 0 instances are not colored and instead contain a small light-gray dash, which makes it immediately clear which cells have confusions and which do not (Figure 6B). Second, Neo can scale the color of the matrix by everything except the diagonal, giving the full color range exclusively to the confusions (the diagonal is removed from the visualization in this case). Third, practitioners can choose from different normalization schemes, presented in detail in Section 4.1, to see different views of the confusions (**T1**). The default normalization scales cells by the instance count, but Neo supports normalizing by the `rows` or `columns`. We can read recall and precision respectively from the diagonal of the normalized matrix. Normalization is set in the spec in the `normalization` field (Figure 5).

5.3.2 Performance Metrics Per Class. Related to choosing different normalization schemes, respondents from our formative research indicated that confusion matrices lack analysis context for looking at other metrics alongside the visualization. Performance metrics, such as accuracy, precision, recall, and others are not readily accessible from a confusion matrix. Aggregate metrics such as these can also be broken down from the model-level to the class-level to support better class-by-class analysis. Neo solves this problem by visualizing both aggregate and per-class metrics on the right-hand side of the confusion matrix as an additional column per metric (Figure 1A and Figure 8), where the top number corresponds to the aggregate metric, and numbers aligned with each row correspond to each class. Besides the metrics listed above, Neo also includes metrics such as the count of actual and predicted instances, true/false positives, and true/false negatives. These are all set in the spec in the `measures` field (Figure 5). While this addition may seem small, it is one of the most common limitations of conventional confusion matrices and was continuously requested by respondents from our formative research.

5.4 Visualizing Hierarchical Labels

Hierarchical datasets are one of the more complex structures discovered from our formative research that conventional confusion matrices do not support (see Section 4.2). Following our design goal to preserve the confusion matrix representation, Neo supports hierarchical labels (see Figure 3) through multiple design improvements (**T2**). First, the class labels on the axes are nested according to the hierarchy, where classes further in the hierarchy are indented (see Figure 1B and Figure 9). Second, the matrix is partitioned into blocks based on the lowest hierarchy level. Hovering over any cell in the matrix highlights its parent hierarchy indicator black (vertical gray bars) for easier tracking (Figure 9A). Together, these two improvements help users understand model performance with the hierarchy directly represented in the visualization.

Neo interactively collapses sub-hierarchies in two ways (see Figure 9). First, selecting a parent class on either axis toggles between showing or hiding the children classes. The interaction collapses (or expands) the parent class and recomputes the confusion data to accurately represent the new aggregate class category in the matrix (**T2**). This implements a Focus+Context paradigm, by expanding class categories of interest while keeping surrounding categories available nearby [9]. Neo models hierarchies as virtual category trees [31], and expands and collapses sub-matrices symmetrically, since the asymmetric case makes confusion much harder to reason about. Alternatively, selecting the magnifying glass icon triggers a drill-down, replacing the entire visualization with only the selected sub-hierarchy and remaps the color (or size) encoding. These techniques allow practitioners to explore larger confusion matrices by reducing the number of visible classes shown and comparing class categories against one another. Regardless of technique, the spec is also updated to record which sub-hierarchies are collapsed or zoomed, set in the `collapsed` and `filter` fields respectively (Figure 5), ensuring that when returning to Neo in the future, or sharing the current view, a user picks up where they left off (**T4**).

5.5 Visualizing Multi-output Labels

Multi-output labels are another more complex structure discovered from our formative research unsupported by conventional confusion matrices (see Section 4.3). Analyzing multi-output models is difficult since confusions are represented in an unbounded high-dimensional space (see Figure 3), which inhibits directly applying conventional matrix visualization. To preserve the confusion matrix representation familiarity, Neo supports three mechanisms to transform high-dimensional confusions into 2D (**T3**): conditioning, marginalization, and nesting (for details of each, see Section 4.3). Inspired by previous work in exploratory visualization [29, 38], in Neo, visualizing multi-output labels leverages an *interactive shelf* to specify label transformations (Figure 7). The interactive shelf contains all multi-output labels for a given dataset. Multi-output labels are either activated or not; activating a multi-output label toggles its color from blue to gray. Activating a multi-output label displays the label in the confusion matrix for analysis.

Conditioning. The first technique to transform multi-output confusions is conditioning, i.e., analyzing confusions for one label given a class of a different label. In these scenarios, Neo conditions the confusion matrix based on the value of a specified label. A practitioner can select to condition the matrix on an actual or predicted class from the conditioning label in the interactive shelf. Note that when a multi-output label is used for conditioning, it can no longer be used for nesting. Similar to the other techniques, these options are reflected in the spec in the `where` field (Figure 5).

Figure 7: Neo's interactive shelf let's practitioners specify how to transform multi-output labels for visualization. Non-activated (gray) multi-output labels are *marginalized*. Activated (blue) multi-output labels define a *nesting* order. The confusion matrix can also be *conditioned* on the value of a particular label.

Marginalization. To visualize high-dimensional confusions, another technique uses marginalization to sum over all other multi-output labels that a practitioner is not interested in. Therefore, in the interactive shelf in Neo, multi-output labels that are not activated, i.e., grayed out instead of blue, are marginalized automatically. In the spec, activated classes are kept in-sync and saved in the `classes` field (Figure 5).

Nesting. Oftentimes a practitioner wants to inspect several multi-output labels at once. To address this issue, Neo nests multi-output labels under one another. Nesting multi-output labels creates a hierarchical label structure, which Neo already supports, where each class of the child label is replicated across all classes of the parent label. Neo automatically nests multi-output labels when more than one label is activated in the interactive shelf. Reordering the labels in the shelf changes the nesting order. This order is also reflected in the spec as the order of the activated classes in `classes` field (Figure 5).

5.6 System Design and Implementation

Neo is a modern web-based system built with Svelte[3], TypeScript[4], and D3[5]. The spec is implemented as a portable JSON format to easily share confusion matrix configurations with other stakeholders (**T4**). Regarding system scalability, Neo is bounded by conventional SVG constraints in the browser (e.g., displaying tens of thousands of SVG elements). Engineering effort such as leveraging Canvas or WebGL would remove this constraint, however we believe better interactions for configuring confusion matrices to compare relevant classes and submatrices is more helpful to practitioners than rendering the biggest matrix possible.

[3]Svelte: https://svelte.dev
[4]TypeScript: https://www.typescriptlang.org
[5]D3: https://d3js.org

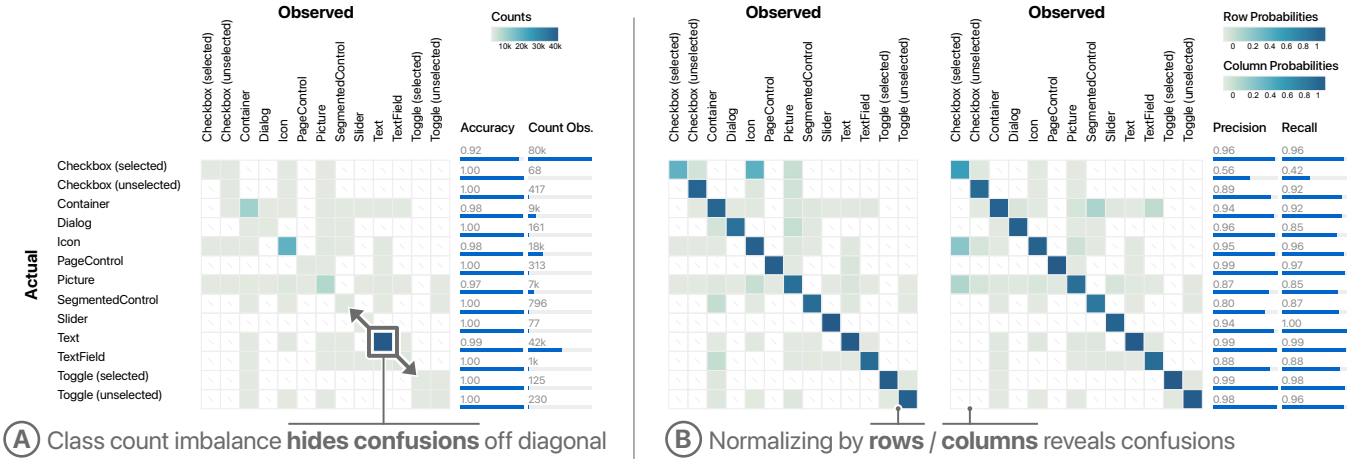

Figure 8: (A) When a dataset has class count imbalance (e.g., some class have many more data instances than others, as seen by the "Count Obs." metric), confusions off the diagonal are hidden and the "Accuracy" metric is misleading. (B) Normalizing by row and/or column probabilities reveals hidden confusions, and has direct connections to other more appropriate model evaluation metrics such as precision and recall.

6 MODEL EVALUATION SCENARIOS

The following three model evaluation scenarios showcase how Neo helps practitioners evaluate models across different domains, including object detection (Section 6.1), large-scale image classification (Section 6.2), and online toxicity detection (Section 6.3).

6.1 Finding Hidden Confusions

Recent work on screen recognition showed how machine learning can create accessibility metadata for mobile applications directly from pixels [41]. An object detection model trained on 77,637 screens extracts user-interface elements from screenshots on-device. The publication includes a confusion matrix for a 13-class classifier that reports and summarizes model performance (test set contains 5,002 instances). With Neo, we can further analyze this confusion matrix and find hidden confusions to help improve the end-user experience of the model.

First, Neo loads the confusion matrix with the default "Accuracy" metric appended on the right-hand side, as seen in Figure 8A. By excluding cells with 0 confusions from the visualization, we can quickly see which class pairs have confusions and which do not. Looking at the accuracies, we see good performance across classes, but in the visualization notice that a few cells dominate the color encoding. When a dataset has strong class count imbalance, e.g., class distribution is not equal, "Accuracy" is a misleading metric to use for evaluating a multi-class model. We confirm this by adding the "Count Observed" metric in the specification to see that the *Text* and *Icon* classes contain many more instances, 42k and 18k respectively (Figure 8A).

With Neo, we normalize the confusion matrix by the row or column probabilities, seen in Figure 8B, that automatically remap the color encoding to reveal hidden confusions. These normalizations are closely related to two other metrics, precision and recall, practitioners use to better inspect performance per class. After adding these metrics to the spec, we see low recall (with row normalization) and low precision (with column normalization) for the *Checkbox (selected)* class; digging into the confusion matrix shows errors with the *Icon* class (Figure 8B, right). We also see confusions between the *Container* class and *SegmentedControl* and *TextField* that were previously hidden in Figure 8A. Neo's design, metrics, and normalization features make error analysis actionable by surfacing hidden error patterns to model builders.

6.2 Traversing Large, Hierarchical Image Classifications

Achieving high accuracy on ImageNet, with its 1.2M+ data instances spread across 1,000 classes, is a standard large-scale benchmark for image classification. Most work considers ImageNet classes as a flat array, but the classes originate from the WordNet [23] hierarchy. To test Neo's scalability, we analyze the results of a ResNet152-V2 [12] deep learning model trained on ImageNet, including its hierarchical structure. The validation set contains 50,000 images.

When loading a large hierarchical confusion matrix, Neo defaults to collapsing all sub-hierarchies and starting at the root. In this configuration, the metrics show the aggregate performance of the entire model, but as we expand into sub-hierarchies these metrics are recomputed per sub-hierarchy and class. Beginning at the root node of the hierarchy, we expand to an early sub-hierarchy titled *object-physical* that contains three sub-categories, each of which we filter for analysis. The first category, *part-portion*, expands fully to contain classes of cloth and towels. The performance on this sub-hierarchy is rather good (strong diagonal), so we continue. Second, the *geological-form* category expands fully to contain classes of natural landscapes (Figure 9A). While the accuracy is high on most classes (above 91-99%), one sub-hierarchy, *shore*, is lower than the others (88%). *Shore* contains two classes, *lakeside* and *seaside*. There are a few confusions between the two, which is expected given their semantic similarity, but there exists another set of confusions between these classes and *sandbar* and *promontory* (point of high

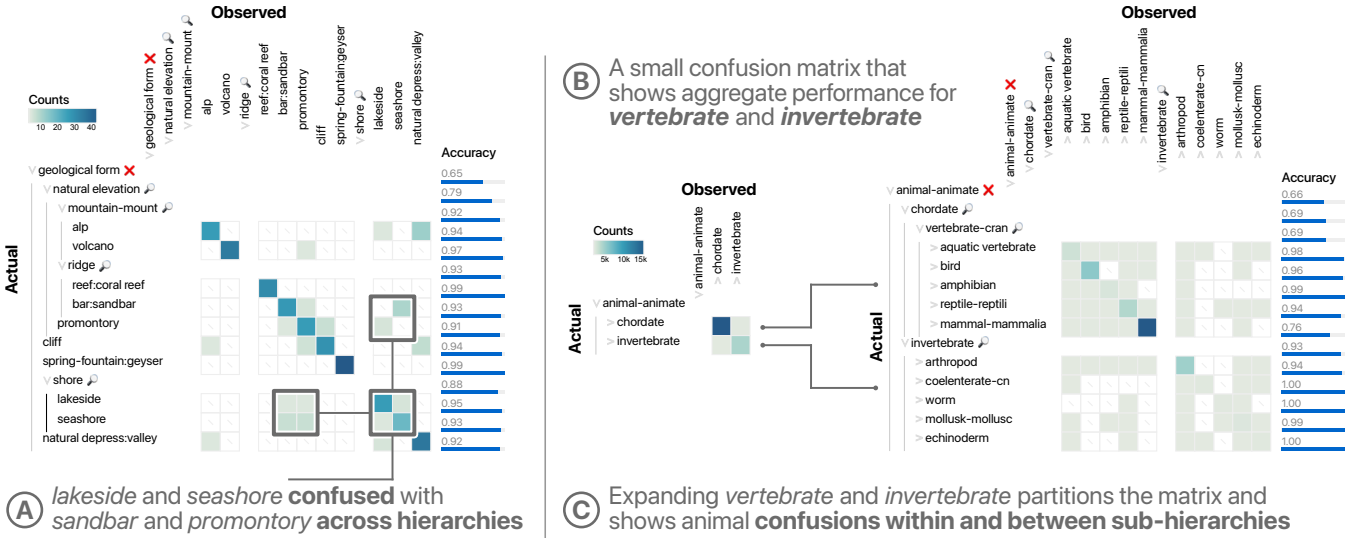

Figure 9: (A) In a deep learning model trained on ImageNet, Neo reveals the *geological form* sub-hierarchy contains confusions between semantically related classes across sub-hierarchies. (B) Another high-level sub-hierarchy for *animal-animate* expands (C) to show detailed confusion comparisons within and between sub-hierarchies of animal classes.

land that juts out into a large body of water), which both belong to a different sub-hierarchy (Figure 9A). Neo enables practitioners to discover these confusions across different sub-hierarchies.

The third and final category, *whole-unit*, in our original sub-hierarchy contains hundreds of classes. We are now interested in inspecting the performance of living things in our model, i.e., the *organism-being* category, which contains four sub-hierarchies: *person-individual*, *plant-flora*, and *fungus* all perform well, but *animal-animate* contains many classes with confusions. Expanding *animal-animate* shows two classes of interest: *chordate-vertebrate* and *invertebrate*, a biological distinction between groups of animals that do or do not have a backbone. This 2x2 confusion matrix is useful for comparing this meaningful, high-level sub-hierarchy (Figure 9B), but expanding both categories one level deeper presents multiple directions for deeper analysis by comparing confusions from animal classes within and between one another (Figure 9C). Throughout this analysis, Neo's specification automatically updates the configuration, so the exact view can be saved and shared with any other project stakeholder.

6.3 Detecting Multi-class and Multi-label Online Toxicity

To make online discussion more productive and respectful, an open challenge tasked developers to build toxicity classifiers [17]. From 159,571 Wikipedia comments labeled by people for toxic behavior, this is a multi-class, multi-label classification problem containing 6 types of non-mutually exclusive behavior: *mild toxicity*, *severe toxicity*, *obscene*, *threat*, *insult*, and *identity hate*; e.g., a comment can be both *mildly toxic* and a *threat*. We analyze the results from a naive one-vs-rest logistic regression classifier from a test set of 47,872 comments.

Neo defaults to loading *mild toxic* comments as the first label to consider. Because this is a multi-output label confusion matrix, Neo visualizes the 2x2 matrix of *mild toxic* comments against *none*, i.e., everything else. The interactive shelf tells us that the other 5 classes are currently marginalized. In Figure 10, left, the "Count Obs." metric tells us this dataset has a large class imbalance, i.e., there are many more non-toxic comments than there are toxic comments. This means our model could struggle with false negatives. Checking this metric indicates that indeed, of the approximately 5k *mild toxic* comments, our naive model only correctly predicts around 2k of them, leaving nearly 3k false negatives. This is an early indication that our model architecture may not suffice for this dataset.

We are interested in other hurtful discussion that could cause emotional harm, therefore we only want to consider *identity hate* comments. To do this, we condition the confusion matrix on *identity hate* (Figure 10A). Figure 10, middle, shows that the model is better at identifying *mild toxic* comments given the instances are also *identity hate*, but there are still *mild toxic* false negatives present.

Beyond *mild toxic*, we now want to inspect more serious comments within *severe toxic*. In Neo, we activate and nest *severe toxic* comments under *mild toxic* comments to consider the occurrence of both multi-output labels simultaneously (Figure 10B). From Figure 10, right, we see the model correctly identifies some of these comments, but suffers a similar problem as *mild toxic* comments in that it has many false negatives. We could consider other confusion matrix configurations, such as nesting *obscene* under *mild toxic* comments to form a larger hierarchy as shown in Figure 1C, but already, we can confidently conclude that our first model cannot distinguish between *mild toxic* comments and benign comments. To improve this model, the next step is likely choosing a different architecture, such as a long short-term memory model, that can learn richer features from the raw text data.

7 DISCUSSION AND FUTURE WORK

Our work opens up many research directions that envision a future where confusion matrices can be further transformed into a powerful and flexible performance analysis tool.

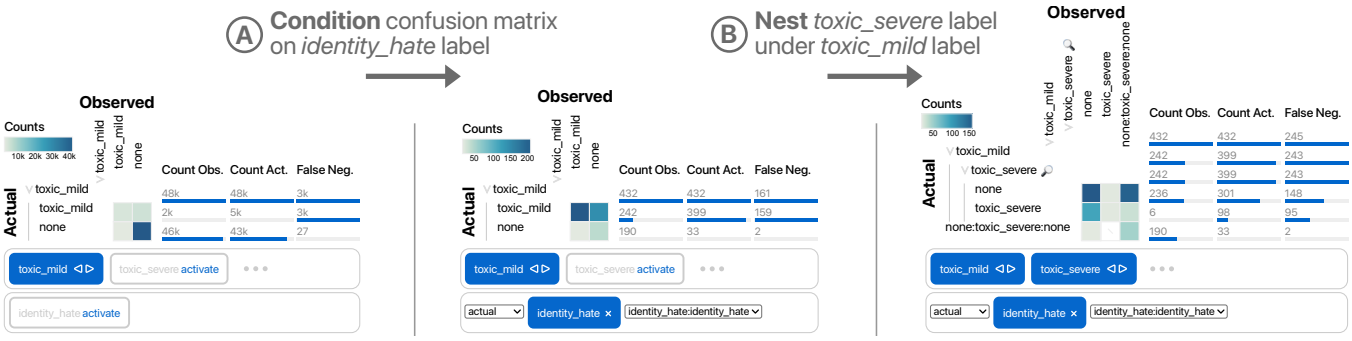

Figure 10: (A) Conditioning a confusion matrix for a toxicity classification model on *identity hate* comments filters all confusions according to the value of the label. (B) Nesting the confusion matrix by *toxic mild* and *toxic severe* allows us to visualize both labels simultaneously.

Confusion Matrix Visualization Scalability. Scaling confusion matrix visualization remains an important challenge. In Neo, hierarchical data can be collapsed to focus on smaller submatrices, since most of the time comparing classes within nearby categories is more meaningful, e.g., comparing "apple" to "orange" instead of "apple" to "airplane." However, in the case of a one-dimensional, large confusion matrix (e.g., more than 100+ classes), the conventional representation suffers from scalability problems. Beyond scrolling and zooming, we envision richer interactions and extensions to our algebra to handle larger confusion matrices. For example, investigating a "NOT" operation that ignores columns to produce a better color mapping to find smaller matrix cells, or leveraging classic table seriation techniques [6].

Automatic Submatrix Discovery. Related to scalability, further algorithmic advancements could help automatically find interesting submatrices of a confusion matrix. From our formative research, this was briefly discussed in the context of a large matrix. Automatically finding groups of cells based on some metric, e.g., low-performing classes, could help guide a practitioner towards important confusions to fix and reduce the number of cells on screen.

Interactive Analysis with Metadata. In data annotation efforts, other metadata is collected besides the raw data features and label(s). How can we use this other metadata to explore model confusions? For example, in an image labelling task, annotators may be asked to draw a bounding box around an object. We could then ask questions about patterns of confusions when metadata such as "bounding box" was small, represented in our confusion matrix algebra as: $P(X, Y \mid \text{bounding box area} < A)$. We could also compute new metrics, such as the percentage of small bounding boxes when the "apple" class was confused with "orange," represented in our confusion matrix algebra as: $P(\text{bounding box area} < A \mid X = \text{apple}, Y = \text{orange})$. Interactive query interfaces that support these types of questions could help practitioners attribute confusions to specific features or metadata, saving time when searching for error patterns.

Comparing Model Confusions Over Time. Machine learning development is an inherently iterative process [4, 13, 15, 24], where multiple models are often compared against each other. Two common comparison scenarios include (1) training multiple models at once, and (2) retraining a model after debugging. In the first scenario, it would be useful to interactively compare confusion matrices against one another to select the best performing model. In the second scenario, using a confusion matrix to compare an improved model against its original version could help practitioners test whether their improvements were successful. One potential comparison technique could be to take the difference between confusion matrices to find the biggest changes.

Creating Datasets from Confusions. While visualizing confusion matrices and aggregate errors helps practitioners debug their models, it can be useful to inspect individual data instances. From our formative research, practitioners expressed interest in extracting instances from confusion matrix cells. Future interactions such as filtering by confusion type and previewing instances within each cell could support extracting and creating new subsets of data for future error analysis.

8 CONCLUSION

From our formative research, one respondent reported that *"confusion matrices are one type of analysis when analyzing performance... doing thorough analysis requires looking at lots of different distributions of the data."* This quote raises a keen point that while confusion matrices remain a ubiquitous visualization for model evaluation, they are only one view into model behavior. There is no one-size-fits-all model evaluation visualization, nor one magic model metric. Regardless, confusion matrices continue to be an excellent tool to teach modeling fundamentals to novices and an invaluable tool for practitioners building industry-scale systems.

In this work, we generalize the capabilities of confusion matrices while maintaining their familiar representation. Through formative research, we design an algebra that models confusion matrices as probability distributions and expresses more variations of confusion matrices, e.g., datasets with hierarchical and multi-labels. Based on this algebra, we develop Neo, a visual analytics system that allows practitioners to flexibly author, interact with, and share confusion matrices. Lastly, we demonstrate Neo's utility with three model evaluation scenarios that help people better understand model performance and reveal hidden confusions.

ACKNOWLEDGMENTS

We thank our colleagues at Apple for their time and effort integrating our research with their work. We especially thank Lorenz Kern who helped with initial prototyping datasets. Jochen Görtler is supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161.

REFERENCES

[1] Bilal Alsallakh, Allan Hanbury, Helwig Hauser, Silvia Miksch, and Andreas Rauber. 2014. Visual methods for analyzing probabilistic classification data. *IEEE Transactions on Visualization and Computer Graphics* (2014). https://doi.org/10.1109/tvcg.2014.2346660

[2] Bilal Alsallakh, Amin Jourabloo, Mao Ye, Xiaoming Liu, and Liu Ren. 2017. Do convolutional neural networks learn class hierarchy? *IEEE Transactions on Visualization and Computer Graphics* (2017). https://doi.org/10.1109/tvcg.2017.2744683

[3] Bilal Alsallakh, Zhixin Yan, Shabnam Ghaffarzadegan, Zeng Dai, and Liu Ren. 2020. Visualizing classification structure of large-scale classifiers. In *ICML Workshop on Human Interpretability in Machine Learning*.

[4] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. https://doi.org/10.1109/icse-seip.2019.00042

[5] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/2702123.2702509

[6] Jacques Bertin. 1983. *Semiology of graphics*. University of Wisconsin Press.

[7] Daniel Bruckner. 2014. *ML-o-Scope: A diagnostic visualization system for deep machine learning pipelines*. Technical Report. https://doi.org/10.21236/ada605112

[8] Olivier Caelen. 2017. A Bayesian interpretation of the confusion matrix. *Annals of Mathematics and Artificial Intelligence* (2017). https://doi.org/10.1007/s10472-017-9564-8

[9] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. 1999. *Readings in information visualization: Using vision to think*. Morgan Kaufmann Publishers Inc.

[10] Edgar Frank Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM* (1970).

[11] Graham R Gibbs. 2007. Thematic coding and categorizing. *Analyzing Qualitative Data* (2007). https://doi.org/10.4135/9781849208574.n4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer. https://doi.org/10.1007/978-3-319-46493-0_38

[13] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit. 2020. ConfusionFlow: A model-agnostic visualization for temporal analysis of classifier confusion. *IEEE Transactions on Visualization and Computer Graphics* (2020). https://doi.org/10.1109/tvcg.2020.3012063

[14] Robert Hogg and Elliot Tanis. 2020. *Probability and statistical inference*. Pearson.

[15] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and visualizing data iteration in machine learning. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3313831.3376177

[16] Mohammad Hossin and M. N. Sulaiman. 2015. A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process* (2015). https://doi.org/10.5121/ijdkp.2015.5201

[17] Jigsaw. 2017. Toxic comment classification challenge. *Kaggle* (2017).

[18] Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Chau. 2017. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics* (2017). https://doi.org/10.1109/tvcg.2017.2744718

[19] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. 2012. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*. https://doi.org/10.1145/2254556.2254659

[20] Ashish Kapoor, Bongshin Lee, Desney Tan, and Eric Horvitz. 2010. Interactive optimization for steering machine classification. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '10)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1753326.1753529

[21] Mary Beth Kery, Donghao Ren, Fred Hohman, Dominik Moritz, Kanit Wongsuphasawat, and Kayur Patel. 2020. mage: Fluid moves between code and graphical work in computational notebooks. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. https://doi.org/10.1145/3379337.3415842

[22] Damir Krstinić, Maja Braović, Ljiljana Šerić, and Dunja Božić-Štulić. 2020. Multi-label classifier performance evaluation with confusion matrix. In *Computer Science & Information Technology*. AIRCC Publishing Corporation. https://doi.org/10.5121/csit.2020.100801

[23] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* (1995). https://doi.org/10.1145/219717.219748

[24] Kayur Patel, Naomi Bancroft, Steven M Drucker, James Fogarty, Andrew J Ko, and James Landay. 2010. Gestalt: Integrated support for implementation and analysis in machine learning. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*. https://doi.org/10.1145/1866029.1866038

[25] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D Williams. 2016. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics* (2016). https://doi.org/10.1109/tvcg.2016.2598828

[26] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics* (2016). https://doi.org/10.31219/osf.io/mqzyx

[27] Christin Seifert and Elisabeth Lex. 2009. A novel visualization approach for data-mining-related classification. In *2009 13th International Conference Information Visualisation*. IEEE. https://doi.org/10.1109/iv.2009.45

[28] Hong Shen, Haojian Jin, Ángel Alexander Cabrera, Adam Perer, Haiyi Zhu, and Jason I Hong. 2020. Designing alternative representations of confusion matrices to support non-expert public understanding of algorithm performance. *Proceedings of the ACM on Human-Computer Interaction* (2020). https://doi.org/10.1145/3415224

[29] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics* (2002). https://doi.org/10.1109/2945.981851

[30] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Query, analysis, and visualization of hierarchically structured data using polaris. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/775047.775064

[31] Aixin Sun and Ee-Peng Lim. 2001. Hierarchical text classification and evaluation. In *Proceedings 2001 IEEE International Conference on Data Mining*. IEEE. https://doi.org/10.1109/ICDM.2001.989560

[32] Robert Susmaga. 2004. Confusion matrix visualization. In *Intelligent Information Processing and Web Mining*. Springer. https://doi.org/10.1007/978-3-540-39985-8_12

[33] Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S. Tan. 2009. EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, 10 pages. https://doi.org/10.1145/1518701.1518895

[34] Niklas Tötsch and Daniel Hoffmann. 2021. Classifier uncertainty: Evidence, potential impact, and probabilistic treatment. *PeerJ Computer Science* (2021). https://doi.org/10.7717/peerj-cs.398

[35] S. van den Elzen and J. J. van Wijk. 2011. BaobabView: Interactive construction and analysis of decision trees. In *2011 IEEE Conference on Visual Analytics Science and Technology*. https://doi.org/10.1109/vast.2011.6102453

[36] James Wexler. 2017. Facets: An open source visualization tool for machine learning training data. http://ai.googleblog.com/2017/07/facets-open-source-visualization-tool.html.

[37] Leland Wilkinson and Michael Friendly. 2009. The history of the cluster heat map. *The American Statistician* (2009). https://doi.org/10.1198/tas.2009.0033

[38] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3025453.3025768

[39] Qian Yang, Jina Suh, Nan-Chen Chen, and Gonzalo Ramos. 2018. Grounding interactive machine learning tool design in how non-experts actually build models. In *Proceedings of the 2018 Designing Interactive Systems Conference*. https://doi.org/10.1145/3196709.3196729

[40] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert. 2019. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Transactions on Visualization and Computer Graphics* (2019). https://doi.org/10.1109/tvcg.2018.2864499

[41] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. *Proceedings of the CHI Conference on Human Factors in Computing Systems* (2021). https://doi.org/10.1145/3411764.3445186